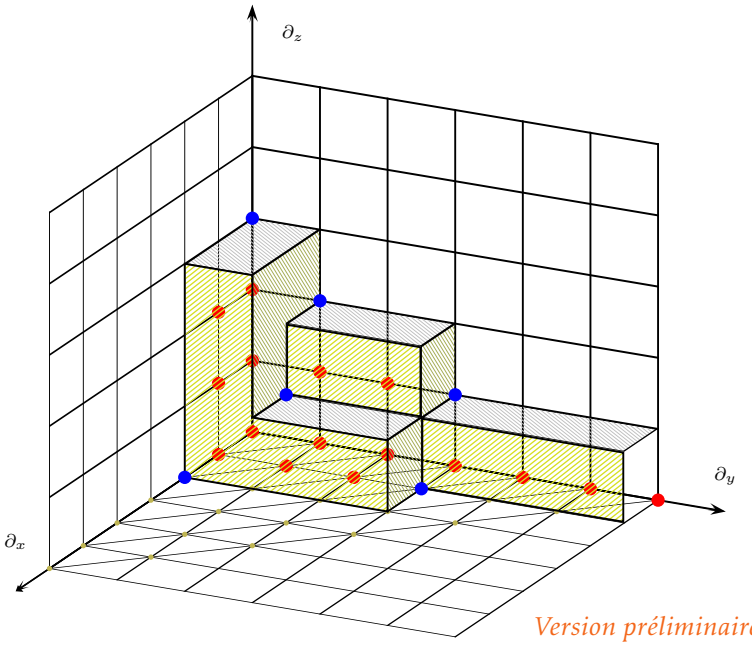
Algorithmes Efficaces en Calcul Formel

Alin Bostan, Frédéric Chyzak, Marc Giusti, Romain Lebreton, Grégoire Lecerf, Bruno Salvy, Eric Schost

Algorithmes Efficaces en Calcul Formel

Alin Bostan
Frédéric Chyzak
Marc Giusti
Romain Lebreton
Grégoire Lecerf
Bruno Salvy
Éric Schost



Version préliminaire du 11 janvier 2017

Alin Bostan
INRIA,
Université Paris-Saclay

Frédéric Chyzak
INRIA,
Université Paris-Saclay

Marc Giusti
CNRS, UMR 7161,
École polytechnique,
Laboratoire d'Informatique

Romain Lebreton
Université de Montpellier,
Laboratoire d'informatique, de robotique et de microélectronique de Montpellier,
CNRS, UMR 5506, CC477

Grégoire Lecerf
CNRS, UMR 7161,
École polytechnique,
Laboratoire d'Informatique

Bruno Salvy
INRIA,
École Normale Supérieure de Lyon,
Laboratoire de l'Informatique du Parallélisme,
CNRS, UMR 5668

Éric Schost
University of Waterloo,
Cheriton School of Computer Science

Publié par les auteurs. Site web : https://hal.archives-ouvertes.fr/AECF/.

Le style LaTeX utilisé est une adaptation de The Legrand Orange Book disponible à l'url http://www.LaTeXTemplates.com/.

Version préliminaire du 11 janvier 2017.

Table des matières

Avant-propos

Le calcul formel traite des objets mathématiques exacts d'un point de vue informatique. Cet ouvrage « Algorithmes efficaces en calcul formel » explore deux directions : la calculabilité et la complexité. La calculabilité étudie les classes d'objets mathématiques sur lesquelles des réponses peuvent être obtenues algorithmiquement. La complexité donne ensuite des outils pour comparer des algorithmes du point de vue de leur efficacité.

Les trois premières parties passent en revue l'algorithmique efficace sur les objets fondamentaux que sont les entiers, les polynômes, les matrices, les séries et les solutions d'équations différentielles ou de récurrences linéaires. Nous y montrons que de nombreuses questions portant sur ces objets admettent une réponse en complexité optimale ou quasi-optimale, en insistant sur les principes généraux de conception d'algorithmes efficaces.

La quatrième partie est dédiée à la factorisation des polynômes. Il s'agit d'un des premiers problèmes d'une importance majeure en calcul formel, pour lequel il existe encore plusieurs questions ouvertes conséquentes. Nous y présentons les algorithmes classiques de factorisation séparable, sans carré et irréductible. Nous traitons tout d'abord le cas des polynômes à une variable dont les coefficients sont dans un corps finis puis des nombres rationnels. Le cas des polynômes à plusieurs variables vient ensuite et nous nous concentrons essentiellement sur le cas de deux variables pour les résultats de complexité.

La cinquième partie aborde les systèmes d'équations polynomiales. Il s'agit de montrer comment répondre à des questions de nature géométrique posées sur les solutions de ces systèmes, tant par des approches à base de récriture (bases standard ou de Gröbner) que plus directement liées à la structure de l'ensemble décrit (résolution géométrique).

La sixième et dernière partie traite de l'intégration et de la sommation définie. De nombreux calculs d'intégrales et de sommes de fonctions ou de suites spéciales de

la combinatoire ou de la physique mathématique peuvent être abordés algorithmiquement. Pour résoudre ces problèmes efficacement, presque tous les algorithmes abordés au cours des cinq premières parties sont utilisés.

Nous n'avons pas cherché à écrire un document de référence pour le calcul formel. Au contraire, nous avons souvent choisi de mettre l'accent sur les idées clés, quitte à présenter sous forme d'exercices les aspects les plus techniques, et à renvoyer le lecteur à des ouvrages ou articles de recherche qui contiennent les solutions. Nous détaillons un grand nombre de résultats très récents qui conduisent souvent le lecteur à la frontière des connaissances. Ainsi, une portion importante des parties III à VI n'était à notre connaissance jamais apparue dans un livre.

Il y a aussi des thèmes classiques du calcul formel que nous n'abordons pas, comme la recherche de primitives (algorithme de Risch) ou de solutions liouvilliennes d'équations différentielles. Nous préférons en effet privilégier un point de vue général où les équations sont vues comme des opérateurs servant à représenter des solutions.

Les prérequis mathématiques nécessaires à la lecture sont modestes et de nombreuses définitions classiques sont rappelées et illustrées au fur et à mesure des besoins. Le texte est par conséquent accessible tant à des étudiants en mathématiques qu'en informatique.

Nous espérons pouvoir être utiles à tous les étudiants et jeunes ou moins jeunes chercheurs et enseignants en mathématiques effectives, ou plus spécifiquement en calcul formel, désireux d'approfondir leurs connaissances ou à la recherche de matériaux pédagogiques.

Cet ouvrage est une synthèse de notes de cours rédigées principalement pour le cours du même nom que nous avons donné pendant plus de dix ans au *Master Parisien de Recherche en Informatique* de l'Université Paris Diderot, des Écoles Normales Supérieures de Cachan et de Paris, et de l'École polytechnique. La partie concernant les systèmes polynomiaux provient aussi de notes de cours donnés au DEA *Méthodes algébriques* puis au *Master Algèbre et Géométrie* de l'Université Pierre-et-Marie-Curie, mais aussi au DEA *Informatique Mathématique et Applications* de l'École Normale Supérieure de Paris, l'École polytechnique, l'Université Pierre-et-Marie-Curie, l'Université Paris Diderot, et l'Université Paris-Sud. Plusieurs parties ont aussi fait l'objet de mini-cours plus spécialisés donnés à l'occasion des *Journées Nationales de Calcul Formel* en 2007, 2010, 2011, et 2013.

Des versions préliminaires de parties de ce livre ont bénéficié des commentaires de relecteurs que nous remercions ici : Philippe Dumas, Claude-Pierre Jeannerod, Marc Mezzarobba, Pierre Nicodème, Anne Vaugon, Gilles Villard.

Ouvrages de référence

Pour les prérequis en informatique, sur les modèles de calcul et la théorie de la complexité, ainsi que sur les algorithmes de base, nous renvoyons le lecteur aux livres de Aho, Hopcroft, Ullman [AHU74], de Knuth [Knu97], de Cormen, Leiserson, Rivest et Stein [Cor+09], et de Stern [Ste94]. Quant aux prérequis en mathématiques, ils concernent essentiellement des points classiques d'algèbre, couverts par exemple par le livre de Lang [Lan02].

Les références générales sur les algorithmes du calcul formel sont deux livres : celui de von zur Gathen et Gerhard [GG03] et celui, plus élémentaire, de Geddes, Czapor et Labahn [GCL92]. La complexité est également utilisée comme fil conducteur dans le livre de Bini et Pan [BP94a] ainsi que dans celui, plus difficile, de Bürgisser, Clausen et Shokrollahi [BCS97]. Mentionnons que les ouvrages, de Geddes, Czapor, et Labahn [GCL92], de Zippel [Zip93], et de Yap [Yap00], couvrent aussi des sujets absents ici comme les manipulations d'expressions symboliques, l'arithmétique des polynômes à plusieurs variables, l'intégration symbolique, ou encore le calcul des racines des polynômes à une variable à coefficients rationnels.

Les aspects calculatoires en algèbre linéaire sont très vastes et il nous a fallu nous restreindre, en deuxième partie, à couvrir seulement les quelques algorithmes nécessaires aux parties suivantes. Les livres de Pan [Pan01 ; Pan84b], et d'Abdeljaoued et Lombardi [AL04] peuvent être utilisés comme références sur ce sujet.

Une bonne introduction à notre cinquième partie, sur les systèmes polynomiaux, est le livre de Cox, Little et O'Shea [CLO96]. Parmi les ouvrages sur les bases de Gröbner, le livre de Becker et Weispfenning [BW93] reste une référence très complète. Un point de vue plus mathématique est développé dans le livre d'Eisenbud [Eis95].

L'algorithmique des corps finis est assez peu présente ici. Ce sujet intervient désormais davantage dans les cours de cryptographie et codes correcteurs. Les livres récents de Shoup [Sho09] et Joux [Jou09] fournissent des contenus très aboutis dans cette direction.

De premiers éléments pour aborder notre troisième partie, sur les équations différentielles et les récurrences linéaires, sont donnés dans le livre de Petkovšek, Wilf et Zeilberger [PWZ96]. Enfin, le livre de Saito, Sturmfels et Takayama [SST00] couvre, en différentiel, certains aspects de la théorie de l'élimination dans le cadre non commutatif de notre sixième partie.

Bibliographie

AHU74 Aho, Alfred V., John E. Hopcroft et Jeffrey D. Ullman (1974). *The design and analysis of computer algorithms*. Addison-Wesley Publishing Co.

AL04 Abdeljaoued, J. et H. Lombardi (2004). *Méthodes matricielles : introduction à la complexité algébrique*. Vol. 42. Mathématiques & Applications. Springer-Verlag.

BCS97 Bürgisser, Peter, Michael Clausen et M. Amin Shokrollahi (1997). *Algebraic complexity theory*. Vol. 315. Grundlehren der Mathematischen Wissenschaften. Springer-Verlag.

BP94a Bini, Dario et Victor Y. Pan (1994). *Polynomial and matrix computations. Vol. 1*. Progress in Theoretical Computer Science. Fundamental algorithms. Birkhäuser Boston, Inc., Boston, MA,

BW93 Becker, Thomas et Volker Weispfenning (1993). *Gröbner bases. A computational approach to commutative algebra*. Vol. 141. Graduate Texts in Mathematics. Springer-Verlag New York.

CLO96 Cox, David, John Little et Donal O'Shea (1996). *Ideals, varieties, and algorithms. An introduction to computational algebraic geometry and commutative algebra*. 2e éd. Undergraduate Texts in Mathematics. Springer-Verlag.

Cor+09 Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest et Clifford Stein (2009). *Introduction to algorithms*. Third. MIT Press, Cambridge, MA.

Eis95 Eisenbud, David (1995). *Commutative algebra. With a view toward algebraic geometry*. Vol. 150. Graduate Texts in Mathematics. Springer-Verlag New York.

GCL92 Geddes, Keith O., Stephen R. Czapor et George Labahn (1992). *Algorithms for computer algebra*. Kluwer Academic Publishers.

GG03 Gathen, Joachim von zur et Jürgen Gerhard (2003). *Modern computer algebra*. 2e éd. Cambridge University Press.

Jou09 Joux, Antoine (2009). *Algorithmic cryptanalysis*. Chapman & Hall/CRC Cryptography and Network Security Series. Chapman et Hall/CRC.

Knu97 Knuth, Donald E. (1997). *The art of computer programming*. 3e éd. Vol. 2 : Seminumerical Algorithms. Computer Science and Information Processing. Addison-Wesley Publishing Co.

Lan02 Lang, Serge (2002). *Algebra*. 3e éd. Vol. 211. Graduate Texts in Mathematics. Springer-Verlag.

Pan01 Pan, Victor Y. (2001). *Structured matrices and polynomials*. Unified superfast algorithms. Birkhäuser Boston Inc.

Pan84b Pan, Victor (1984). *How to multiply matrices faster*. Vol. 179. Lecture Notes in Computer Science. Springer-Verlag.

PWZ96 Petkovšek, Marko, Herbert S. Wilf et Doron Zeilberger (1996). A = B. A. K. Peters.

Sho09 Shoup, Victor (2009). *A computational introduction to number theory and algebra*. 2e éd. Cambridge University Press.

SST00 Saito, Mutsumi, Bernd Sturmfels et Nobuki Takayama (2000). *Gröbner deformations of hypergeometric differential equations*. Springer-Verlag.

Ste94 Stern, Jacques (1994). *Fondements mathématiques de l'informatique*. Ediscience international.

Yap00 Yap, Chee (2000). *Fundamental problems in algorithmic algebra*. Oxford University Press.

Zip93 Zippel, Richard (1993). *Effective polynomial computation*. Kluwer Academic Publishers.

1. Calcul formel et complexité

Résumé

Le calcul formel étudie dans quelle mesure un ordinateur peut « faire des mathématiques », et pour celles qu'il peut faire, à quelle vitesse. Ce chapitre introductif présente d'abord les problèmes de calculabilité. Le point de départ est le théorème plutôt négatif de Richardson–Matiyasevich. Ensuite, un survol rapide des thèmes qui seront abordés tout au long de cet ouvrage montre que malgré ce théorème, bien des questions peuvent être traitées algorithmiquement. La seconde partie du chapitre aborde la théorie de la complexité, pose les conventions et les définitions principales, ainsi que les résultats de base sur la méthodologie « diviser pour régner ».

1.1 Décider, calculer

Fondements logiques

D'une certaine manière, le calcul formel est fondé sur une contrainte d'origine logique.

Théorème 1.1 — Richardson–Matiyasevich. Dans la classe des expressions formées à partir d'une variable X et de la constante 1 par les opérations d'anneau $+, -, \times$ et la composition avec la fonction $\sin(\cdot)$ et la fonction valeur absolue $|\cdot|$, le test d'équivalence à 0 est indécidable.

Autrement dit, il n'existe pas d'algorithme permettant pour toute expression de cette classe de déterminer en temps fini si elle vaut 0 ou non. Plus généralement tout test d'égalité peut bien entendu se ramener à tester l'égalité à zéro dès que la soustraction existe. Cette limitation de nature théorique explique la difficulté et

parfois la frustration que rencontrent les utilisateurs débutants des systèmes de calcul formel face à des fonctions de « simplification », qui ne peuvent être qu'heuristiques.

Pour effectuer un calcul, il est pourtant souvent crucial de déterminer si des expressions représentent 0 ou non, en particulier pour évaluer une fonction qui possède des singularités (comme la division). L'approche du calculateur formel expérimenté consiste à se ramener autant que faire se peut à des opérations d'un domaine dans lequel le test à zéro est décidable. Le calcul formel repose ainsi de manière naturelle sur des constructions algébriques qui préservent la décidabilité du test à 0. En particulier, les opérations courantes sur les vecteurs, matrices, polynômes, fractions rationnelles, ne nécessitent pas d'autre test à 0 que celui des coefficients. La notion d'effectivité permet de préciser ce point de vue.

Définition 1.1 Une structure algébrique (groupe, anneau, corps, espace vectoriel,...) est dite *effective* si l'on dispose :
- d'une structure de données pour en représenter les éléments ;
- d'algorithmes pour en effectuer les opérations et pour y tester l'égalité et autres prédicats.

Par exemple, dans un anneau effectif, outre l'égalité, les opérations requises sont l'addition, la soustraction et la multiplication. Pour une structure ordonnée, des algorithmes sont aussi requis pour effectuer les comparaisons. Cette notion d'effectivité est bien évidemment attachée à un modèle de calcul. Dans cet ouvrage nous considérerons principalement les *machines à accès direct* (voir ci-dessous). Néanmoins il est connu que les fonctions calculables par ces machines sont les mêmes que pour les machines de Turing (mais avec des temps d'exécution bien sûr différents).

Structures et constructions de base

Les objets les plus fondamentaux sont assez faciles à représenter en machine de manière exacte. Nous considérons tour à tour les plus importants d'entre eux, en commençant par les plus élémentaires. Ils s'assemblent ensuite à l'aide de tableaux ou de listes pour en former de plus complexes.

Entiers machine

Les entiers fournis par les processeurs sont des entiers modulo une puissance de 2 (le nombre de bits d'un mot machine, typiquement 32 ou 64). Ils sont appelés des *entiers machine*. Les opérations rendues disponibles par le processeur sont l'addition, la soustraction, la multiplication et parfois la division. La norme ANSI du langage C fournit au programmeur la division et le modulo pour ces entiers, c'est-à-dire que le compilateur implante ces opérations si le processeur ne le fait pas.

Entiers

Pour manipuler des entiers dont la taille dépasse celle d'un mot machine, il est commode de les considérer comme écrits dans une base B assez grande :

$$\mathrm{N} = a_0 + a_1 \mathrm{B} + \cdots + a_k \mathrm{B}^k.$$

L'écriture est unique si l'on impose $0 \leq a_i < \mathrm{B}$. (Le signe est stocké séparément.) Ces nombres peuvent être stockés dans des tableaux d'entiers machine. Les objets obtenus

sont des entiers de taille arbitraire appelés parfois *bignums*.

L'addition et le produit peuvent alors être réduits à des opérations sur des entiers inférieurs à B^2, au prix de quelques opérations de propagation de retenue. Le choix de B dépend un peu du processeur. Si le processeur dispose d'une instruction effectuant le produit de deux entiers de taille égale à celle d'un mot machine, renvoyant le résultat dans deux mots machines, alors B pourra être pris aussi grand que le plus grand entier tenant dans un mot machine. Sinon, c'est la racine carrée de ce nombre qui sera utilisée pour B. Quoiqu'il en soit, nous avons donc le résultat suivant :

Lemme 1.2 L'anneau $\mathbb{Z}$ des entiers relatifs est effectif.

Entiers modulaires

Les calculs avec des polynômes, des fractions rationnelles ou des matrices à coefficients entiers souffrent souvent d'une maladie propre au calcul formel : la croissance des expressions intermédiaires. Les entiers produits comme coefficients des expressions intervenant lors du calcul sont de taille disproportionnée par rapport à ceux qui figurent dans l'entrée et dans la sortie.

Exemple 1.1 Voici le déroulement typique du calcul du plus grand diviseur commun (pgcd) de deux polynômes à coefficients entiers par l'algorithme d'Euclide (Chapitre 6), où nous notons rem(.,.) le reste de la division :

$$
\begin{aligned}
P_0 &= 7X^5 - 22X^4 + 55X^3 + 94X^2 - 87X + 56,\\
P_1 &= 62X^4 - 97X^3 + 73X^2 + 4X + 83,\\
P_2 &= \mathrm{rem}(P_0, P_1) = \frac{113293}{3844}X^3 + \frac{409605}{3844}X^2 - \frac{183855}{1922}X + \frac{272119}{3844},\\
P_3 &= \mathrm{rem}(P_1, P_2) = \frac{18423282923092}{12835303849}X^2 - \frac{15239170790368}{12835303849}X + \frac{10966361258256}{12835303849},\\
P_4 &= \mathrm{rem}(P_2, P_3) = -\frac{216132274653792395448637}{44148979404824831944178}X - \frac{631179956389122192280133}{88297958809649663888356},\\
P_5 &= \mathrm{rem}(P_3, P_4) = \frac{20556791167692068695002336923491296504125}{3639427682941980248860941972667354081}.
\end{aligned}
$$

Les coefficients de ces polynômes intermédiaires font intervenir des entiers qui croissent de manière exponentielle, alors que le résultat recherché est 1.

Grâce à la division euclidienne, on obtient immédiatement.

Lemme 1.3 Pour tout entier $n \geq 2$, l'anneau $\mathbb{Z}/n\mathbb{Z}$ des entiers modulo n est effectif.

Ces entiers modulaires remédient à ce problème de croissance intermédiaire de deux manières. D'une part, pour un calcul de décision, de dimension, ou de degré, l'exécution de l'algorithme sur la réduction de l'entrée modulo un nombre premier donne un algorithme *probabiliste* répondant à la question. Cette technique peut aussi servir de base à un algorithme *déterministe* lorsque les nombres premiers pour lesquels la réponse est fausse peuvent être maîtrisés. Nous verrons comment exploiter cette

idée pour le calcul du pgcd dans le Chapitre 6.

D'autre part, les entiers modulaires sont utilisés dans les algorithmes reposant sur le théorème des restes chinois. Ce théorème indique qu'un entier inférieur au produit de nombres premiers $p_1 \cdots p_k$ peut être reconstruit à partir de ses réductions modulo $p_1, \ldots, p_k$. Lorsqu'une borne sur la taille du résultat est disponible, il suffit d'effectuer le calcul modulo suffisamment de nombres premiers (choisis en pratique assez grands pour que leur nombre soit faible et assez petits pour que les opérations tiennent dans un mot machine), pour ensuite reconstruire le résultat, court-circuitant de la sorte toute croissance intermédiaire.

Vecteurs et matrices

Une fois donnée une représentation exacte pour des coefficients, il est facile de construire des vecteurs ou matrices comme des tableaux, ou plus souvent comme des tableaux de pointeurs sur les coefficients. Les opérations de produit par un scalaire, de produit de matrices ou de produit d'une matrice par un vecteur se réduisent aux opérations d'addition et de multiplication sur les coefficients. Il en va de même pour la recherche de noyau ou d'inverse de matrices.

Proposition 1.4 Si $\mathbb{K}$ est un corps effectif, l'espace vectoriel $\mathbb{K}^n$ l'est aussi, ainsi que l'anneau $\mathcal{M}_n(\mathbb{K})$.

Polynômes

Les polynômes peuvent être représentés de plusieurs manières, et la meilleure représentation dépend des opérations que l'on souhaite effectuer. Pour un polynôme en une variable, les choix principaux sont :

— la représentation dense : comme pour les entiers, le polynôme est représenté comme un tableau de (pointeurs sur les) coefficients ;
— la représentation creuse : le polynôme est représenté comme une liste de paires (coefficient, exposant) généralement triée par les exposants.

Dans les deux cas, nous avons clairement la propagation de l'effectivité aux anneaux de polynômes :

Proposition 1.5 Si $\mathbb{A}$ est un anneau effectif, alors $\mathbb{A}[X]$ l'est aussi.

L'usage itéré de cette proposition fournit les polynômes à plusieurs variables.

Fractions rationnelles

Les rationnels peuvent être stockés comme des paires où numérateur et dénominateur sont des entiers de taille arbitraire. Les opérations d'addition et de multiplication se réduisent aux opérations analogues sur les entiers et le test d'égalité à zéro se réduit au test d'égalité à 0 sur le numérateur. De même, les fractions rationnelles sont représentées par des paires de polynômes. Les opérations d'addition, produit, division, se réduisent aux additions et multiplications sur les coefficients. Plus généralement, nous obtenons le résultat suivant pour les corps de fractions rationnelles :

Proposition 1.6 Si $\mathbb{A}$ est un anneau intègre effectif, alors son corps des fractions est effectif.

Séries tronquées

Les séries tronquées

$$\sum_{k=0}^{N} a_k X^k + O(X^{N+1})$$

se représentent pratiquement comme des polynômes. La différence principale apparaît lors du produit : les coefficients des termes d'exposant au moins $N+1$ n'ont pas besoin d'être calculés, ni stockés. La structure considérée alors est l'anneau quotient $\mathbb{A}[X]/(X^{N+1})$ obtenu en faisant le quotient de l'anneau $\mathbb{A}[X]$ par l'idéal qu'y engendre X^{N+1}. Les éléments de ce quotient sont représentés par le $(N+1)$-uplet de coefficients $(a_0,\ldots,a_N)$. Le test à 0 revient à tester que ces coefficients sont tous nuls.

Proposition 1.7 Si $\mathbb{A}$ est un anneau effectif et $N \in \mathbb{N}$, alors $\mathbb{A}[X]/(X^{N+1})$ est un anneau effectif.

Cette structure de données joue un rôle très important non seulement pour des calculs d'approximations, mais aussi comme une représentation *exacte*. En voici trois exemples :

1. Une fraction rationnelle dont les numérateurs et dénominateurs ont un degré borné par d peut être reconstruite à partir d'un développement en série à l'ordre $2d+1$. Cette représentation joue ainsi un rôle clé dans la manipulation des nombres algébriques (Chapitre 3), et dans le calcul efficace de la division euclidienne de polynômes, de suites récurrentes linéaires (comme le calcul rapide du 10 000^{e} nombre de Fibonacci au Chapitre 4) et du polynôme minimal d'une matrice creuse (Chapitre 9).
2. Il est possible de reconstruire une équation différentielle linéaire à coefficients polynomiaux à partir du développement en série d'une solution et de bornes sur l'ordre et le degré des coefficients. De façon analogue, il est possible de reconstruire une récurrence linéaire à coefficients polynomiaux à partir des premières valeurs d'une de ses solutions. L'outil algorithmique pour effectuer ces calculs de *devinette* (*guessing* en anglais) est le calcul rapide d'approximants de Padé–Hermite (Chapitre 7).
3. Un polynôme en deux variables peut être reconstruit à partir du développement en série d'une solution. L'efficacité de la factorisation des polynômes à deux variables, ou encore de la résolution de systèmes polynomiaux par la méthode dite de *résolution géométrique*, abordée au Chapitre 28 repose de manière cruciale sur cette opération, qui doit être effectuée rapidement.

Équations comme structures de données

Une fois construits les objets de base que sont les polynômes, les séries ou les matrices, il est possible d'aborder des objets mathématiques construits *implicitement*. Ainsi, il est bien connu qu'il n'est pas possible de représenter toutes les solutions

de polynômes de haut degré par radicaux, mais de nombreuses opérations sur ces solutions sont aisées en prenant le polynôme lui-même comme structure de données. Ce point de vue permet d'étendre le domaine d'application du calcul formel pourvu que des algorithmes soient disponibles pour effectuer les opérations souhaitées (typiquement addition, multiplication, multiplication par un scalaire, test d'égalité) par manipulation des équations elles-mêmes.

Nombres algébriques

C'est ainsi que l'on nomme les solutions de polynômes à une variable. Le résultat est spectaculaire.

Proposition 1.8 Si $\mathbb{K}$ est un corps effectif, alors sa clôture algébrique $\bar{\mathbb{K}}$ l'est aussi.

Les opérations d'addition et de multiplication peuvent être effectuées à l'aide de résultants (Chapitre 6). Ceux-ci peuvent être calculés efficacement à l'aide de séries (Chapitre 3). La division s'obtient par l'algorithme d'Euclide sur les polynômes (Chapitre 6), et le test d'égalité se déduit du pgcd. Par exemple, il est possible de prouver assez facilement une identité comme

$$\frac{\sin\frac{2\pi}{7}}{\sin^2\frac{3\pi}{7}} - \frac{\sin\frac{\pi}{7}}{\sin^2\frac{2\pi}{7}} + \frac{\sin\frac{3\pi}{7}}{\sin^2\frac{\pi}{7}} = 2\sqrt{7} \tag{1.1}$$

une fois que l'on reconnaît qu'il s'agit d'une égalité entre nombres algébriques.

Systèmes polynomiaux

Vu l'importance des systèmes polynomiaux en calcul formel, une grande partie de cet ouvrage leur sera consacrée. Un résultat majeur de cette partie est le suivant.

Proposition 1.9 Soient $\mathbb{K}$ un corps effectif, et $f_1,\dots,f_p$ des polynômes de l'anneau $\mathbb{K}[X_1,\dots,X_n]$. Alors l'anneau quotient $\mathbb{K}[X_1,\dots,X_n]/(f_1,\dots,f_n)$ est effectif.

Ce quotient est un outil de base pour répondre à de nombreuses questions naturelles sur un système de polynômes, comme l'existence de solutions, la dimension de l'espace des solutions (qui indique s'il s'agit d'une surface, d'une courbe, ou de points isolés), le degré, ou le calcul d'une paramétrisation de l'ensemble des solutions.

Il est également possible d'éliminer une ou des variables entre des polynômes. Cette opération s'interprète géométriquement comme une projection (Chapitre 25). Dans le cas le plus simple, elle permet de calculer un polynôme s'annulant sur les abscisses des intersections de deux courbes (Chapitre 6). Une autre application est le calcul d'une équation d'une courbe donnée sous forme paramétrée.

Équations différentielles linéaires

Cette structure de données permet de représenter de nombreuses fonctions usuelles (exponentielle, fonctions trigonométriques et trigonométriques hyperboliques, leurs réciproques), ainsi que de nombreuses fonctions spéciales de la physique mathématique (fonctions de Bessel, de Struve, d'Anger, ..., fonctions hypergéométriques

et hypergéométriques généralisées), ainsi bien sûr que de multiples fonctions auxquelles n'est pas attaché un nom classique. Les opérations d'addition et de produit sont effectuées par des variantes noncommutatives du résultant qui se ramènent à de l'algèbre linéaire élémentaire (Chapitre 14). Le test à zéro se réduit à tester l'égalité d'un nombre fini de conditions initiales. Une partie de ces résultats se résume comme suit :

Proposition 1.10 Si $\mathbb{K}$ est un corps effectif, les séries formelles de $\mathbb{K}[[X]]$ qui sont solutions d'équations différentielles linéaires à coefficients dans $\mathbb{K}[X]$ forment un anneau effectif.

En d'autres termes, des structures de données finies permettent de manipuler ces objets infinis et d'en tester l'égalité ou la nullité.

Ainsi, des identités élémentaires comme $\sin^2 X + \cos^2 X = 1$ sont non seulement facilement prouvables algorithmiquement, mais elles sont également calculables, c'est-à-dire que le membre droit se calcule à partir du membre gauche.

Les relations étroites entre équations différentielles linéaires et récurrences linéaires — les séries solutions des unes ont pour coefficients les solutions des autres — aboutissent aux mêmes réponses algorithmiques que les questions analogues sur des suites. Par exemple, l'identité de Cassini sur les nombres de Fibonacci

$$F_{n+2}F_n - F_{n+1}^2 = (-1)^{n+1}, \qquad n \geq 0$$

est exactement du même niveau de difficulté que $\sin^2 X + \cos^2 X = 1$. Le pendant du résultat précédent est donc la proposition suivante :

Proposition 1.11 Si $\mathbb{K}$ est un corps effectif, l'ensemble des suites de $\mathbb{K}^{\mathbb{N}}$ solutions de récurrences linéaires à coefficients dans $\mathbb{K}[n]$ forme un anneau effectif.

Systèmes d'équations différentielles et de récurrences linéaires

Ces systèmes sont aux équations différentielles ou de récurrences ce que les systèmes polynomiaux sont aux polynômes en une variable. Les mêmes opérations sont disponibles. En particulier, l'élimination s'étend dans ce cadre en introduisant des algèbres d'opérateurs adaptés (Chapitre 31). Une application très importante, le *téléscopage créatif*, permet de calculer automatiquement des sommes et des intégrales définies. Ainsi,

$$\sum_{k=0}^{n}\left(\sum_{j=0}^{k}\binom{n}{j}\right)^3 = n2^{3n-1} + 2^{3n} - 3n2^{n-2}\binom{2n}{n},$$

$$\sum_{n=0}^{\infty} H_n(x)H_n(y)\frac{u^n}{n!} = \frac{\exp\left(\frac{4u(xy-u(x^2+y^2))}{1-4u^2}\right)}{\sqrt{1-4u^2}},$$

$$\frac{1}{2}J_0(x)^2 + J_1(x)^2 + J_2(x)^2 + \cdots = \frac{1}{2},$$

$$\int_{-1}^{+1} \frac{e^{-px}\mathrm{T}_n(x)}{\sqrt{1-x^2}}\,dx = (-1)^n \pi \mathrm{I}_n(p),$$

$$\int_0^{+\infty} x e^{-px^2} \mathrm{J}_n(bx)\mathrm{I}_n(cx)\,dx = \frac{1}{2p}\exp\left(\frac{c^2-b^2}{4p}\right)\mathrm{J}_n\left(\frac{bc}{2p}\right),$$

$$\int_0^{+\infty} x\mathrm{J}_1(ax)\mathrm{I}_1(ax)\mathrm{Y}_0(x)\mathrm{K}_0(x)\,dx = -\frac{\ln(1-a^4)}{2\pi a^2},$$

$$\sum_{k=0}^{n} \frac{q^{k^2}}{(q;q)_k (q;q)_{n-k}} = \sum_{k=-n}^{n} \frac{(-1)^k q^{(5k^2-k)/2}}{(q;q)_{n-k}(q;q)_{n+k}},$$

sont des formules qui mettent en jeu diverses fonctions spéciales ou polynômes orthogonaux classiques, et qui peuvent être prouvées automatiquement. Les algorithmes correspondants seront décrits au Chapitre 32. L'énoncé en termes d'anneaux effectifs est un peu lourd et omis ici.

En conclusion, les exemples ci-dessus illustrent bien la manière dont le calcul formel parvient à effectuer de nombreux calculs utiles dans les applications malgré l'indécidabilité révélée par le théorème de Richardson et Matiyasevich.

1.2 Calculer rapidement

En pratique, la calculabilité n'indique que la faisabilité. Il faut disposer d'algorithmes efficaces et des bonnes implantations pour pouvoir effectuer des calculs de grande taille. La première partie de cet ouvrage est consacrée aux algorithmes efficaces sur les structures de base du calcul formel. L'efficacité sera mesurée par la théorie de la complexité et nous ferons ressortir des principes récurrents dans la conception d'algorithmes efficaces.

Exemple 1.2 Pour donner une idée de ce que veut dire rapidement, voici ce qui peut être calculé en *une seconde* avec le système Maple sur un ordinateur portable d'aujourd'hui [a], en notant $\mathbb{K}$ le corps $\mathbb{Z}/p\mathbb{Z}$ à p éléments, $p = 67\,108\,879$ étant un nombre premier de 26 bits (dont le carré tient sur un mot machine) :

1. Entiers :
   - — produit de deux entiers avec 30 000 000 chiffres ;
   - — factorielle de 1 300 000 (environ 7 000 000 chiffres) ;
   - — factorisation d'un entier de 42 chiffres (produit de deux nombres premiers de taille la moitié).
2. Polynômes dans $\mathbb{K}[X]$:
   - — produit de deux polynômes de degré 650 000 ;
   - — pgcd et résultant de deux polynômes de degré 12 500 ;
   - — factorisation d'un polynôme de degré 170 (produit comme ci-dessus).
3. Polynômes dans $\mathbb{K}[X, Y]$:
   - — résultant de deux polynômes de degré total 20 (sortie de degré 400) ;

— factorisation d'un polynôme de degré 160 en deux variables.

4. Matrices :
   — produit de deux matrices 850×850 à coefficients dans $\mathbb{K}$;
   — déterminant d'une matrice $1\,400 \times 1\,400$ à coefficients dans $\mathbb{K}$;
   — polynôme caractéristique d'une matrice 500×500 à coefficients dans $\mathbb{K}$;
   — déterminant d'une matrice 200×200 dont les coefficients sont des entiers 32 bits.

Ces exemples montrent qu'il est relativement aisé de calculer avec des objets de taille colossale. Dans la plupart de ces exemples, les algorithmes naïfs mettraient plusieurs années pour le même calcul. On voit aussi qu'en une seconde, les algorithmes sont déjà dans leur régime asymptotique et qu'une analyse asymptotique suffira à obtenir des informations précises. Tout ceci donne envie d'introduire une mesure de complexité des différents algorithmes permettant d'expliquer, voire de prédire, les différences entre les tailles atteintes pour ces questions.

a. Ce texte est écrit en 2015. Les ordres de grandeurs sont les mêmes sur différents ordinateurs et pour les différents systèmes de calcul formel principaux, avec des forces et des faiblesses différentes.

Mesures de complexité

Pour bien définir la complexité, il faut se donner : un modèle de machine ; les opérations disponibles sur cette machine ; leur coût unitaire. La complexité en espace mesure la mémoire utilisée par l'exécution de l'algorithme, et la complexité en temps, la somme des coûts unitaires des opérations effectuées par l'algorithme. Dans cet ouvrage, nous n'aborderons pas la complexité en espace.

Machine à accès direct

Le modèle que nous utiliserons est celui de la *machine à accès direct* (MAD), appelée aussi *Random Access Machine* (RAM) en anglais. Dans ce modèle, un programme lit et écrit des entiers sur deux bandes différentes et utilise un nombre arbitraire de registres entiers pour ses calculs intermédiaires. Les opérations élémentaires (l'assembleur de la machine) sont la lecture, l'écriture (sur bande ou en registre), l'addition, la soustraction, le produit, la division et trois instructions de saut : saut inconditionnel, saut si un registre est nul et saut si un registre est positif. Un point technique est que le programme ne fait pas partie des données, il n'est donc pas modifiable. Cet ouvrage ne rentre jamais dans les subtilités de l'utilisation de ce modèle, qui ne sert qu'à fixer précisément les mesures utilisées.

Complexité arithmétique ou binaire

Nous considérerons deux mesures de complexité :

1. En calcul formel, de nombreux algorithmes peuvent être décrits comme opérant de façon abstraite sur une structure algébrique $\mathbb{A}$ effective. D'une façon concrète, la représentation des éléments de $\mathbb{A}$ importe peu et l'utilisation de chaque opération élémentaire de $\mathbb{A}$ (opération arithmétique ou prédicat) peut être parfaitement identifiée au cours de l'exécution de l'algorithme.
   La *complexité arithmétique* d'un tel algorithme est définie comme le nombre total d'opérations arithmétiques et de tests de prédicats effectués dans $\mathbb{A}$. Cette mesure ne prend volontairement pas en compte les opérations de copie, celles

sur les compteurs de boucle, les indirections, etc. En pratique cette mesure reflète souvent le coût réel de l'algorithme si le coût des opérations dans $\mathbb{A}$ est prépondérant et si chaque opération requiert un temps essentiellement constant. C'est le cas par exemple si $\mathbb{A}$ est un corps fini. Ce n'est en revanche pas le cas si $\mathbb{A}$ est le corps des nombres rationnels $\mathbb{Q}$ et que la taille de ces nombres croît de façon significative durant les calculs.

2. Pour étudier le coût des algorithmes opérant sur des entiers, la complexité arithmétique avec $\mathbb{A} = \mathbb{Z}$ n'est pas pertinente. Il convient alors de décomposer les entiers dans une base B, qui est en pratique une puissance de 2 fixée (par exemple 2^{32}). Chaque entier est alors vu comme un vecteur d'éléments de $\{0, \ldots, \mathrm{B}-1\}$. Nous appellerons *complexité binaire* la complexité arithmétique associée à $\mathbb{A} = \{0, \ldots, \mathrm{B}-1\}$, muni des tests d'égalité et comparaisons, ainsi que des opérations arithmétiques modulo B. En pratique cette mesure de complexité reflète en général bien les temps observés pour les algorithmes opérant sur des nombres entiers ou rationnels. Néanmoins, précisons que cette mesure de complexité ne correspond pas à celle utilisée classiquement sur une machine de Turing, puisqu'elle néglige les coûts d'accès mémoire. Par exemple le coût binaire pour transposer une matrice de taille $n \times n$ à coefficients dans $\mathbb{A} = \{0, \ldots, \mathrm{B}-1\}$ est nul avec notre définition : aucune opération arithmétique n'est requise.

La notation $\mathrm{O}(\cdot)$

Nous utilisons la notation $\mathrm{O}(\cdot)$ pour exprimer une borne sur la complexité des algorithmes. La signification précise de la notation

$$f(n) = \mathrm{O}(g(n)), \qquad n \to \infty$$

est qu'il existe $\mathrm{K} > 0$ et $\mathrm{A} > 0$ tels que pour tout $n > \mathrm{A}$, f et g sont liés par l'inégalité

$$|f(n)| \leq \mathrm{K}|g(n)|.$$

Lorsque plusieurs paramètres interviennent dans l'analyse de la complexité, il faut absolument préciser lequel tend vers l'infini pour que cette notation ait un sens. Si plusieurs d'entre eux tendent vers l'infini, soit ils sont liés par des inégalités qui seront précisées, soit la définition ci-dessus s'étend avec une constante K qui ne dépend d'aucun des paramètres.

Afin de simplifier les bornes de complexité nous utiliserons parfois la notation $\tilde{\mathrm{O}}(\cdot)$ de sorte à cacher des facteurs logarithmiques. Plus précisément, nous écrirons

$$f(n) = \tilde{\mathrm{O}}(g(n)), \qquad n \to \infty$$

lorsque qu'il existe un entier $k \geq 0$ tel que

$$f(n) = \mathrm{O}(g(n) \log_2^k(\max(|g(n)|, 2))).$$

La notation $\mathrm{O}(\cdot)$ intervient aussi dans cet ouvrage pour représenter la troncature des séries. L'expression

$$f(\mathrm{X}) := g(\mathrm{X}) + \mathrm{O}(\mathrm{X}^{\mathrm{N}})$$

signifiera que le polynôme ou la série g est tronqué après son N-ième terme et que le résultat, un polynôme, est stocké dans f.

Exemple 1.3 L'addition de deux entiers de n bits demande donc $O(1)$ opérations arithmétiques et $O(n)$ opérations binaires (quelque soit la base B, puisque la différence entre les bases joue sur la constante cachée dans le $O(\cdot)$).

Exemple 1.4 Le calcul de $n!$ par la méthode naïve requiert n opérations arithmétiques et $O(n^2 \log^2 n)$ opérations binaires. Le choix de la base B n'intervient que dans la constante cachée dans le $O(\cdot)$. Nous verrons au Chapitre 15 qu'il est possible d'abaisser ce coût à seulement $O(n^{1/2} \log n)$ opérations arithmétiques dans $\mathbb{Z}$, et $O(n \log^3 n)$ opérations binaires. Les idées utilisées dans les algorithmes rapides permettant d'atteindre ces complexités fournissent aussi le meilleur algorithme connu de factorisation déterministe d'entiers et des algorithmes très efficaces pour le calcul de millions de décimales de π, $\log 2$ et de nombreuses autres constantes.

Taille

Un algorithme et une structure de données sont généralement dotés d'une notion naturelle de taille et il s'agit d'étudier le coût de l'algorithme en fonction de cette taille. Pour simplifier, il est souvent commode de considérer le comportement asymptotique de ce coût lorsque la taille tend vers l'infini. Il est important de comprendre que la complexité d'un problème n'a de sens qu'une fois la structure de données fixée pour l'entrée comme pour la sortie.

Par exemple, pour les polynômes, le choix de la représentation dense conduit à mesurer la complexité par rapport au degré, alors que le choix de la représentation creuse met en avant le nombre de monômes. Pour la factorisation, la complexité est polynomiale en le degré, mais exponentielle en le nombre de monômes, dans le cas le pire.

Cas le pire, cas moyen

La complexité dans le cas le pire est le maximum des complexités pour toutes les entrées d'une taille donnée. C'est celle que nous étudierons. Il est souvent utile de considérer aussi la complexité en moyenne, lorsque l'on peut mettre une mesure sur l'ensemble des entrées de taille bornée. Pour la plupart des algorithmes que nous étudierons dans la première partie de cet ouvrage, il n'y a pas de différence importante entre les deux. Ce n'est plus le cas en revanche pour la complexité des algorithmes de factorisation, ou pour ceux qui opèrent sur les systèmes polynomiaux.

Bornes inférieures

La recherche de bornes inférieures de complexité est très difficile. Par exemple, à l'heure actuelle on ne sait pas prouver que la multiplication de matrices est nécessairement plus coûteuse qu'un nombre borné d'additions. Dès qu'il est possible de montrer que tous les bits de l'entrée doivent être pris en compte, la somme de la taille de l'entrée et de la taille de la sortie est une borne inférieure sur la complexité. En effet, dans le modèle MAD, chacune des écritures et des lectures prend une opération.

Définition 1.2 Si N est la somme de la taille de l'entrée et de la taille de la sortie, un algorithme sera dit *quasi-optimal* lorsque sa complexité est bornée par $\tilde{O}(N)$.

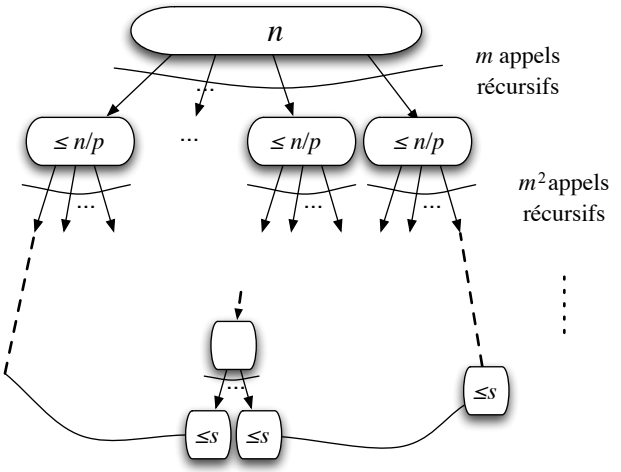


FIGURE 1.1 – Appels récursifs d'un algorithme « diviser pour régner ».

L'essentiel de la première partie de cet ouvrage consistera à rechercher des algorithmes quasi-optimaux pour les opérations de base sur les structures de données fondamentales.

Diviser pour régner

Le principe le plus important de la conception d'algorithmes efficaces est la méthodologie « diviser pour régner ». Il consiste à résoudre un problème en le réduisant à un certain nombre m d'entrées de tailles divisées par p (le plus souvent $p = 2$) puis à recombiner les résultats (voir Figure 1.1). Le coût de la recombinaison et éventuellement du découpage préliminaire est borné par une fonction T de la taille des entrées. Lorsque les entrées ont une taille inférieure à p ou suffisamment petite, notée s, un autre algorithme de coût κ indépendant de n est invoqué. Le coût total obéit alors souvent à une récurrence de la forme

$$\mathrm{C}(n) \leq \begin{cases} \mathrm{T}(n) + m\mathrm{C}(\lceil n/p \rceil), & \text{si } n \geq s (\geq p) \\ \kappa & \text{sinon.} \end{cases} \tag{1.2}$$

La notation $\lceil x \rceil$ désigne l'entier k tel que $k - 1 < x \leq k$ (le « plafond » de x).

Qualitativement, le coût total de cette approche dépend fortement de la fonction T. Lorsque T est relativement élevée, les premiers niveaux de l'arbre contribuent à l'essentiel du coût et, à une constante près, le coût est donc dominé par la première étape de récursion. Les algorithmes à base d'itération de Newton du Chapitre 3 sont de ce type. À l'inverse, pour une fonction T assez faible, le bas de l'arbre domine le coût qui sera proportionnel au nombre de feuilles [1] $\mathrm{O}((n/s)^{\log_p m})$. L'algorithme de

1. La notation $\log_p x$ représente le logarithme en base p de x, c'est-à-dire $\log x / \log p$. L'identité facile à vérifier $a^{\log_p b} = b^{\log_p a}$ sera utile.

multiplication de polynômes de Karatsuba du Chapitre 2 et l'algorithme de Strassen pour la multiplication de matrices (Chapitre 8) rentrent dans cette catégorie. Enfin, il est possible que tous les $O(\log_p(n/s))$ niveaux de l'arbre contribuent de manière assez équilibrée, menant à un coût en $O(T(n)\log n)$. La transformée de Fourier rapide (Chapitre 2) et le classique algorithme de tri fusion sont dans cette dernière catégorie.

Un cadre commode pour nos applications est résumé dans le lemme suivant, dont une version simplifiée que nous utiliserons dans la plupart des cas est donnée ensuite par le Théorème 1.13. Nous commençons par le cas simple où la taille est une puissance de p, où il est possible de donner des inégalités précises, ce qui prépare le passage au cas général et à l'asymptotique.

Lemme 1.12 — Diviser pour régner. Soit C une fonction obéissant à l'inégalité (1.2) avec $m > 0$, $\kappa > 0$ et T une fonction telle que

$$T(pn) \geq qT(n), \qquad n \in \mathbb{N}, \tag{1.3}$$

avec $q > 1$. Alors, si n est une puissance positive de p,

$$C(n) \leq \begin{cases} \left(1 - \frac{m}{q}\right)^{-1} T(n) + \kappa n^{\log_p m} & \text{si } q > m, \\ T(n)\log_p n + \kappa n^{\log_p q} & \text{si } q = m, \\ n^{\log_p m}\left(\frac{T(n)}{n^{\log_p q}}\frac{q}{m-q} + \kappa\right) & \text{si } q < m. \end{cases}$$

Le premier cas est celui où la complexité est portée par le haut de l'arbre, dans le deuxième elle l'est par tous les niveaux à parts égales, et dans le dernier par les bas niveaux.

Démonstration. L'utilisation répétée de l'inégalité (1.2) sur C donne

$$\begin{aligned} C(n) &\leq T(n) + mC\left(\frac{n}{p}\right), \\ &\leq T(n) + mT\left(\frac{n}{p}\right) + \cdots + m^{k-1}T\left(\frac{n}{p^{k-1}}\right) + m^k C\left(\frac{n}{p^k}\right), \qquad (1.4) \\ &\leq T(n)\left(1 + \frac{m}{q} + \cdots + \left(\frac{m}{q}\right)^{k-1}\right) + m^k\kappa, \qquad (1.5) \end{aligned}$$

où la dernière ligne résulte de (1.3) et du choix de $k = \lfloor \log_p \frac{n}{s} \rfloor + 1 \in [\log_p \frac{n}{s}, \log_p n]$. Ce choix entraine la suite d'inégalités suivante :

$$1 \leq x \Rightarrow x^k \leq x \cdot x^{\log_p(n/s)} = n^{\log_p x} \cdot x^{1-\log_p s} \leq n^{\log_p x} \tag{1.6}$$

qui permet de borner le deuxième terme de (1.5) avec $x = m$.

Si $m < q$, la somme entre parenthèses est majorée par la série géométrique. Si $m = q$,

la somme comporte k termes égaux. Si $m > q$, récrire la somme sous la forme

$$\left(\frac{m}{q}\right)^{k-1}\left(1+\frac{q}{m}+\cdots+\left(\frac{q}{m}\right)^{k-1}\right)$$

montre que le premier terme de (1.5) est majoré par

$$\mathrm{T}(n)\left(\frac{m}{q}\right)^{k}\frac{q}{m-q}.$$

Ensuite, l'inégalité (1.6) avec $x = m/q$ permet de conclure. ■

En pratique la régularité de T est souvent plus forte que celle donnée par (1.3) et on ne cherche qu'à comparer des estimations de $\mathrm{O}(\cdot)$. Le résultat prend alors une forme plus simple, et sans restriction sur n qui n'a plus besoin d'être une puissance de p.

Théorème 1.13 — Diviser pour régner. Soit C une fonction obéissant à l'inégalité (1.2) avec $m > 0$ et $\kappa > 0$, et soit T une fonction *croissante* telle qu'existent q et r avec $1 < q \leq r$ vérifiant

$$q\mathrm{T}(n) \leq \mathrm{T}(pn) \leq r\mathrm{T}(n), \qquad \text{pour tout } n \text{ assez grand.} \tag{1.7}$$

Alors, lorsque $n \to \infty$

$$\mathrm{C}(n) = \begin{cases} \mathrm{O}(\mathrm{T}(n)), & \text{si } q > m, \\ \mathrm{O}(\mathrm{T}(n)\log n), & \text{si } q = m, \\ \mathrm{O}\left(n^{\log_p m}\dfrac{\mathrm{T}(n)}{n^{\log_p q}}\right) & \text{si } q < m. \end{cases}$$

L'hypothèse (1.7) est en particulier vérifiée par les fonctions courantes $n^\alpha \log^\beta n$ pour les valeurs $\alpha > 0$.

Démonstration. Soit N la puissance de p telle que $n \leq \mathrm{N} < pn$. En déroulant l'inégalité (1.2) on obtient une inégalité similaire à (1.4) où la division par p est remplacée par l'opération $x \mapsto \lceil x/p \rceil$. La fonction T étant supposée croissante, toutes ces valeurs de T sont majorées par les valeurs en N/p^i, $i = 0, \ldots, k-1$ et on obtient donc une majoration (1.5) où n est remplacé par N. Le lemme précédent s'applique alors pour majorer cette expression, majoration qu'il ne reste plus qu'à majorer à son tour.

D'abord, par récurrence, l'inégalité (1.3) entraîne $\mathrm{T}(n) \geq q^{\log_p n}\mathrm{T}(1) = n^{\log_p q}\mathrm{T}(1)$, ce qui permet d'absorber les termes en κ dans les $\mathrm{O}()$. Ensuite, la croissance de T et l'hypothèse (1.7) donnent pour n assez grand $\mathrm{T}(\mathrm{N}) \leq \mathrm{T}(pn) = \mathrm{O}(\mathrm{T}(n))$, ce qui permet de conclure. ■

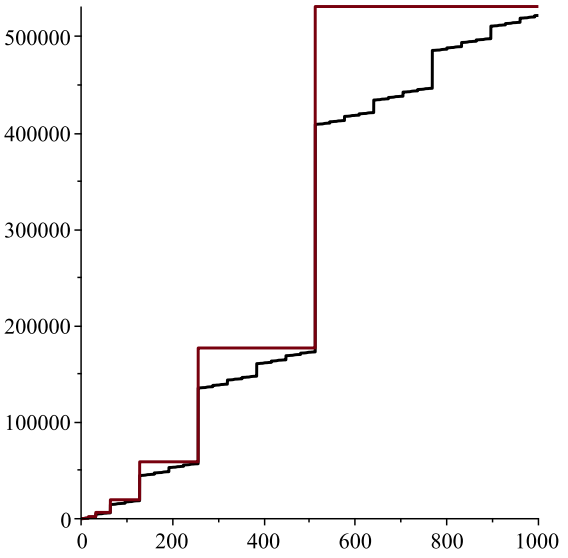


FIGURE 1.2 – Borne sur la complexité de l'algorithme de Karatsuba : en noir, la suite solution de $C(n) = 4n+3C(\lceil n/2\rceil)$, $C(1) = 1$; en rouge, la borne obtenue à l'Exemple 1.5.

Exemple 1.5 La complexité de l'algorithme de Karatsuba du Chapitre 2 vérifie la récurrence

$$K(n) \leq 4n + 3K(\lceil n/2\rceil), \qquad K(1) = 1.$$

Le Lemme 1.12 et le Théorème 1.13 s'appliquent avec $m = 3$, $p = s = 2$, $\kappa = 1$ et T linéaire, si bien que $q = r = 2$, d'où une complexité $K(n) = O(n^{\log_2 3})$, et plus précisément

$$K(n) \leq n^{\log_2 3}\left(\frac{4n}{n}\frac{2}{1} + 1\right) = 9n^{\log_2 3}$$

pour n une puissance de 2, et donc en majorant par la valeur en la puissance de 2 immédiatement supérieure, le cas général est borné par

$$K(n) \leq 9 \cdot 2^{\lceil \log_2 n\rceil \log_2 3} = 9 \cdot 3^{\lceil \log_2 n\rceil}.$$

Cette borne est assez fine, comme le montre la Figure 1.2.

Conventions

Pour toute la suite de cet ouvrage, et sauf mention contraire, nous adoptons les notations et terminologies suivantes :

— tous les anneaux et corps sont supposés effectifs, commutatifs et unitaires ;
— les estimations de complexité sont en nombre d'opérations arithmétiques dans

l'anneau de base ;
— le symbole ■ marque la fin d'une démonstration ;
— les logarithmes sont en base 2, lorsque la base n'est pas spécifiée ;
— le lemme « diviser pour régner » fait référence à l'énoncé du Lemme 1.12 ;
— le théorème « diviser pour régner » fait référence à l'énoncé du Théorème 1.13.

Notes

Le théorème de Richardson–Matiyasevich que nous énonçons a ses origines dans les travaux sur le 10e problème de Hilbert portant sur l'existence d'un algorithme permettant de trouver les racines entières des polynômes multivariés à coefficients entiers. Une étape importante dans la résolution de ce problème est due à Davis, Putnam et Robinson [DPR61] qui montrent en 1961 l'indécidabilité de ce problème si en plus des polynômes on considère la fonction $x \mapsto 2^x$. Quelques années plus tard, Richardson [Ric68] ramène les problèmes de racines réelles en plusieurs variables à ces questions, et, à l'aide d'un codage astucieux par le sinus, y ramène aussi les problèmes en une variable. Par rapport à notre énoncé, son théorème fait intervenir la fonction exp et la constante $\ln 2$ (pour utiliser le résultat de Davis, Putnam et Robinson), ainsi que la constante π (pour utiliser la nullité du sinus aux multiples *entiers* de π). Deux ans plus tard, en 1970, Matiyasevich prouve que la fonction $x \mapsto 2^x$ elle-même peut s'exprimer à l'aide de polynômes, concluant ainsi la preuve de l'indécidabilité du 10e problème de Hilbert. Le théorème de Richardson peut alors être simplifié, et c'est ce que fait Matiyasevich dans son livre [Mat93], où il montre aussi comment se débarrasser de la constante π. Il faut noter que par contre, si l'on remplace le sinus par l'exponentielle, alors la simplification devient décidable [Ric69].

Ces théorèmes s'appliquent à des fonctions. Pour des constantes, l'approche la plus récente réduit le test à zéro à une conjecture de théorie des nombres due à Schanuel qui exprime que les seules relations entre exponentielles et logarithmes sont celles qui découlent des formules d'addition et de multiplication. Si l'on accepte cette conjecture de Schanuel, alors un résultat relativement récent de Macintyre et Wilkie en 1996 [MW96] entraîne l'existence d'un algorithme de reconnaissance de 0 pour la classe des constantes obtenues à partir de 1 par addition, soustraction, multiplication et exponentielle. Un tel algorithme, à base d'évaluation numérique et de LLL (voir Chapitre 20) a ensuite été donné en 1997, à nouveau par Richardson [Ric97].

Les différents modèles de complexité (machine MAD, *straight-line program*, machine de Turing, …) sont bien présentés par Aho, Hopcrof et Ullman dans leur livre [AHU74], ou par Stern [Ste94]. La jolie identité (1.1) est tirée de [Bec+05]. Les résultats de complexité autour de l'algorithmique du « diviser pour régner » sont nombreux. Le premier remonte sans doute à Bentley, Haken et Saxe [BHS80] qui font apparaître la série géométrique. Une version moderne tenant compte des planchers et des plafonds est présentée par Cormen, Leiserson, Rivest et Stein qui l'appellent *Master Theorem* [Cor+09]. Des variantes sophistiquées sont connues [DS11 ; Rou01]. Dans cette direction, Yap [Yap11] donne un théorème prêt à l'emploi dans le cas où plusieurs divisions avec des valeurs de p distinctes ont lieu.

Bibliographie

AHU74 Aho, Alfred V., John E. Hopcroft et Jeffrey D. Ullman (1974). *The design and analysis of computer algorithms*. Addison-Wesley Publishing Co.

Bec+05 Beck, Matthias, Bruce C. Berndt, O-Yeat Chan et Alexandru Zaharescu (2005). « Determinations of analogues of Gauss sums and other trigonometric sums ». In : *International Journal of Number Theory*, vol. 1, n°3, p. 333–356.

BHS80 Bentley, Jon Louis, Dorothea Haken et James B. Saxe (1980). « A general method for solving divide-and-conquer recurrences ». In : *SIGACT News*, vol. 12, n°3, p. 36–44.

Cor+09 Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest et Clifford Stein (2009). *Introduction to algorithms*. Third. MIT Press, Cambridge, MA.

DPR61 Davis, Martin, Hilary Putnam et Julia Robinson (1961). « The decision problem for exponential diophantine equations ». In : *Annals of Mathematics. Second Series*, vol. 74, p. 425–436.

DS11 Drmota, Michael et Wojciech Szpankowski (2011). « A master theorem for discrete divide and conquer recurrences ». In : *SODA'11 : ACM-SIAM Symposium on Discrete Algorithms*. San Francisco, CA : SIAM, p. 342–361.

Mat93 Matiyasevich, Yuri V. (1993). *Hilbert's tenth problem*. Foundations of Computing Series. Translated from the 1993 Russian original by the author, with a foreword by Martin Davis. MIT Press.

MW96 Macintyre, Angus et A. J. Wilkie (1996). « On the decidability of the real exponential field ». In : *Kreiseliana*. A. K. Peters, p. 441–467.

Ric68 Richardson, Daniel (1968). « Some undecidable problems involving elementary functions of a real variable ». In : *Journal of Symbolic Logic*, vol. 33, n°4, p. 514–520.

Ric69 — (1969). « Solution of the identity problem for integral exponential functions ». In : *Zeitschrift fur mathematischen Logik und Grundlagen der Mathematik*, vol. 15, p. 333–340.

Ric97 — (1997). « How to recognize zero ». In : *Journal of Symbolic Computation*, vol. 24, n°6, p. 627–645.

Rou01 Roura, Salvador (2001). « Improved master theorems for divide-and-conquer recurrences ». In : *Journal of the Association for Computing Machinery*, vol. 48, n°2, p. 170–205.

Ste94 Stern, Jacques (1994). *Fondements mathématiques de l'informatique*. Ediscience international.

Yap11 Yap, Chee (2011). « A real elementary approach to the master recurrence and generalizations ». In : *Theory and applications of models of computation*. Vol. 6648. Lecture Notes in Computer Science. Tokyo, Japan : Springer-Verlag, p. 14–26.

Polynômes et séries

2. Multiplication rapide

Résumé

Les algorithmes rapides de calcul du produit de polynômes et d'entiers sont au cœur de l'algorithmique efficace en calcul formel. La plupart des gains de complexité dans les chapitres ultérieurs reposent sur l'efficacité de la multiplication. Pour multiplier deux polynômes de degré n à coefficients dans un anneau $\mathbb{A}$, la méthode classique requiert $O(n^2)$ opérations dans $\mathbb{A}$. De même, l'algorithme scolaire de multiplication de deux entiers à n chiffres nécessite un nombre d'opérations binaires en $O(n^2)$. Nous présentons dans ce chapitre plusieurs algorithmes de multiplication rapide, dont celui de Karatsuba, de complexité $O(n^{1,59})$, ainsi que ceux utilisant la transformée de Fourier rapide, dont la complexité est quasiment linéaire en n.

2.1 Introduction, résultats principaux

Les problèmes abordés dans ce chapitre concernent la complexité arithmétique de la multiplication des polynômes à une variable et la complexité binaire de la multiplication des entiers. Au vu de l'exemple suivant, il est facile de se convaincre de la similitude des deux questions :

Polynômes Soient à multiplier $3X^2 + 2X + 1$ et $6X^2 + 5X + 4$ dans $\mathbb{Z}[X]$,

$$\begin{aligned}(3X^2 + 2X + 1) \times (6X^2 + 5X + 4)\\ = (3 \cdot 6)X^4 + (3 \cdot 5 + 2 \cdot 6)X^3 + (3 \cdot 4 + 2 \cdot 5 + 1 \cdot 6)X^2 + (2 \cdot 4 + 1 \cdot 5)X + (1 \cdot 4)\\ = 18X^4 + 27X^3 + 28X^2 + 13X + 4.\end{aligned}$$

Nombres entiers Soient à multiplier 321 et 654 en base 10,

$$\begin{aligned}&(3\cdot 10^2+2\cdot 10+1)\times(6\cdot 10^2+5\cdot 10+4)\\&=(3\cdot 6)10^4+(3\cdot 5+2\cdot 6)10^3+(3\cdot 4+2\cdot 5+1\cdot 6)10^2+(2\cdot 4+1\cdot 5)10+(1\cdot 4)\\&\qquad=18\cdot 10^4+27\cdot 10^3+28\cdot 10^2+13\cdot 10+4\\&\qquad\qquad=2\cdot 10^5+9\cdot 10^3+9\cdot 10^2+3\cdot 10+4=209934.\end{aligned}$$

Dans les deux cas, nous avons retranscrit l'algorithme naïf, et la suite des calculs est essentiellement la même, si ce n'est que, dans le cas des entiers, il faut en outre gérer les retenues (dernière égalité de l'exemple). On ne sera donc pas surpris que les résultats obtenus dans les deux cas soient très semblables.

Résultats

Dans toute la suite, $(\mathbb{A},+,\times)$ désignera un anneau commutatif effectif. Tout d'abord, nous considérons la complexité arithmétique ; il s'agit de minimiser le nombre d'opérations $(+,-,\times)$ dans $\mathbb{A}$ pour multiplier des polynômes en degré borné. Les premiers résultats à retenir de ce chapitre sont les suivants.

Théorème 2.1 La multiplication des polynômes de degré au plus n dans $\mathbb{A}[X]$ requiert :
- $O(n^2)$ opérations dans $\mathbb{A}$ par l'*algorithme naïf* ;
- $O(n^{1,59})$ opérations dans $\mathbb{A}$ par l'*algorithme de Karatsuba* ;
- $O(n\log n\log\log n)$ — voire dans certains cas $O(n\log n)$ — opérations dans $\mathbb{A}$, via la *transformée de Fourier rapide (FFT)*.

Ainsi, la multiplication des polynômes peut se faire en un coût arithmétique *quasi-linéaire* en leur degré.

La multiplication des polynômes est omniprésente : les algorithmes de calcul de pgcd (plus grand commun diviseur), de pgcd étendu, de factorisation en une ou plusieurs variables, de composition des séries formelles, d'évaluation multipoint, d'interpolation, font tous intervenir des produits de polynômes.

L'analogie entre les entiers et les polynômes va très loin ; la plupart des réponses apportées dans le cadre de la complexité arithmétique trouvent un équivalent en complexité binaire. Cependant, aucun théorème d'équivalence n'est connu ; il se trouve que les mêmes idées algorithmiques s'adaptent plus ou moins facilement dans les deux cadres. Ainsi, on dispose des résultats suivants dans le modèle binaire.

Théorème 2.2 On peut multiplier des entiers de n chiffres binaires par :
- l'*algorithme naïf* en $O(n^2)$ opérations binaires ;
- l'*algorithme de Karatsuba* en $O(n^{1,59})$ opérations binaires ;
- l'*algorithme de Schönhage–Strassen* en $O(n\log n\log\log n)$ opérations binaires.

Les preuves de ces résultats de complexité binaire sont plus délicates que celles de leurs analogues polynomiaux, à cause des problèmes de gestion des retenues. Aussi dans la suite ne traitons nous d'abord en détail que les versions polynomiales de

ces résultats (Théorème 2.1), pour passer ensuite brièvement en revue le cas entier (Théorème 2.2).

En pratique

Les constantes cachées dans les $O(\cdot)$ sont déterminantes pour l'efficacité pratique de tels algorithmes. Par exemple, lorsque $\mathbb{A}$ est un corps fini de taille « raisonnable » (typiquement, dont les éléments sont représentés sur quelques mots machine), pour le produit de polynômes dans les meilleures implantations actuelles (Magma, NTL, FLINT, Mathemagix) :

— l'algorithme de Karatsuba bat l'algorithme naïf pour des degrés d'environ 20 ;
— les méthodes à base de FFT en $O(n \log n)$ battent l'algorithme de Karatsuba pour des degrés de l'ordre de 100, mais ne peuvent pas être utilisées pour des degrés arbitrairement grands (vient un moment où l'on manque de racines de l'unité, voir plus loin) ;
— l'algorithme de type FFT en $O(n \log n \log\log n)$ est utilisé pour des degrés de l'ordre de quelques dizaines ou centaines de milliers.

Certains problèmes, en cryptologie ou en théorie des nombres, nécessitent de manipuler des polynômes de degré de l'ordre de 100 000, tailles pour lesquelles les algorithmes rapides sont indispensables. Plus fréquemment, des degrés de l'ordre de la centaine ou du millier font partie du quotidien du calculateur formel (au moins dans des calculs intermédiaires).

Ces algorithmes ne sont pas concurrents, mais complémentaires. Les bonnes implantations passent automatiquement à l'algorithme le plus efficace en fonction du degré, et ce parfois même au cours d'un même calcul, lorsqu'un algorithme récursif comme celui de Karatsuba atteint une taille pour laquelle il vaut mieux utiliser l'algorithme naïf.

La situation pour les entiers est similaire, même si l'implantation des algorithmes rapides pour les entiers est bien plus délicate en raison des retenues. Dans les meilleures implantations actuelles (Magma, GMP) :

— l'algorithme de Karatsuba bat l'algorithme naïf pour des nombres de l'ordre de 30 chiffres décimaux ;
— les méthodes à base de FFT (Schönhage–Strassen) gagnent pour des nombres d'environ 3 000 chiffres décimaux.

À nouveau, des entiers de quelques centaines de chiffres sont courants dès que l'on sort des exemples jouets. Au-delà, des problèmes venus de la cryptologie ou de la théorie des nombres demandent de manipuler des nombres de taille colossale (de l'ordre de 100 000 000 chiffres ; il faut 10 Mo pour stocker un tel nombre). Ceci justifie amplement les efforts d'implantation d'algorithmes rapides.

Dans la suite de ce chapitre, on travaille avec des polynômes F et G à coefficients dans un anneau $\mathbb{A}$, ayant un degré au plus $n-1$, formellement

$$F = f_0 + \cdots + f_{n-1}X^{n-1} \quad \text{et} \quad G = g_0 + \cdots + g_{n-1}X^{n-1};$$

le problème est alors de calculer (les coefficients de)

$$H = FG = h_0 + \cdots + h_{2n-2}X^{2n-2}.$$

2.2 Algorithme naïf

L'algorithme naïf consiste à développer le produit, c'est-à-dire à écrire

$$H = FG = \sum_{i=0}^{2n-2} h_i X^i \quad \text{avec} \quad h_i = \sum_{j+k=i} f_j g_k.$$

Ainsi, calculer tous les h_i demande $O(n^2)$ opérations dans $\mathbb{A}$. C'est un algorithme de complexité arithmétique *quadratique*.

Exercice 2.1 Montrer que, pour multiplier deux polynômes de degrés m et n, l'algorithme naïf demande au plus $(m+1)\times(n+1)$ multiplications dans $\mathbb{A}$ et mn additions dans $\mathbb{A}$. ■

Exercice 2.2 Montrer que, pour multiplier deux entiers à n chiffres chacun, l'algorithme naïf demande $O(n^2)$ opérations binaires. ■

Exercice 2.3 Estimer la complexité binaire de la méthode naïve lorsque les polynômes ont degré n et des coefficients entiers bornés en valeur absolue par un entier H. ■

Exercice 2.4 Alexander J. Yee et Shigeru Kondo ont calculé 10^{13} décimales de π sur un PC de bureau en 2011 [a]. Ce type de calcul repose sur la multiplication d'entiers de même taille que le nombre de décimales cherchées. En supposant que leur machine était capable d'effectuer 10^{12} opérations à la seconde, montrer qu'ils n'ont pas utilisé l'algorithme naïf. ■

a. Voir `http://www.numberworld.org/misc_runs/pi-5t/announce_en.html`.

2.3 Algorithme de Karatsuba

Un premier raffinement de l'algorithme naïf repose sur la remarque suivante : il est possible de gagner *une* multiplication pour le produit des polynômes de degré 1, parmi les 4 du produit par l'algorithme quadratique. Soient en effet à multiplier les polynômes

$$F = f_0 + f_1 X \quad \text{et} \quad G = g_0 + g_1 X.$$

Leur produit $H = FG = h_0 + h_1 X + h_2 X^2$ peut être obtenu par une forme d'interpolation sur les points $0, 1, \infty$: en 0 on a $H(0) = h_0 = f_0 g_0$; de même, le coefficient de plus haut degré (qui correspond intuitivement à l'évaluation en l'infini) vaut $h_2 = f_1 g_1$. Enfin, la valeur en 1 vaut $h_0 + h_1 + h_2 = F(1)G(1) = (f_0 + f_1)(g_0 + g_1)$. Ainsi, on obtient $h_1 = (f_0 + f_1)(g_0 + g_1) - h_0 - h_2$ pour seulement une multiplication supplémentaire, donc l'ensemble des coefficients du produit pour 3 multiplications et 4 additions. Quelques additions sont perdues par rapport à l'algorithme naïf, mais le gain d'une multiplication va se transformer en gain dans l'*exposant* de l'algorithme, par application récursive.

En effet, dans le cas général des degrés quelconques, il suffit de scinder F et G en deux et de procéder de la même manière. Si F et G sont de degré au plus $n-1$, avec

Entrée F, G de degrés au plus $n-1$.
Sortie $H = FG$.
1. Si $n = 1$, renvoyer FG.
2. Décomposer F et G selon l'équation (2.1).
3. Calculer $A_1 = F^{(0)}G^{(0)}$ et $A_2 = F^{(1)}G^{(1)}$ récursivement.
4. Calculer $A_3 = F^{(0)} + F^{(1)}$ et $A_4 = G^{(0)} + G^{(1)}$.
5. Calculer $A_5 = A_3A_4$ récursivement.
6. Calculer $A_6 = A_5 - A_1$ et $A_7 = A_6 - A_2$.
7. Renvoyer $A_1 + A_7X^k + A_2X^{2k}$.

Algorithme 2.1 – Multiplication de polynômes par l'algorithme de Karatsuba.

$k = \lceil n/2 \rceil$, on pose

$$F = F^{(0)} + F^{(1)}X^k, \quad G = G^{(0)} + G^{(1)}X^k, \tag{2.1}$$

pour des polynômes $F^{(0)}, F^{(1)}, G^{(0)}, G^{(1)}$ de degrés au plus $k-1$. Le produit $H = FG$ s'écrit

$$H = F^{(0)}G^{(0)} + (F^{(0)}G^{(1)} + F^{(1)}G^{(0)})X^k + F^{(1)}G^{(1)}X^{2k}.$$

Cette méthode est résumée dans l'Algorithme 2.1. Le théorème qui suit établit la complexité de l'algorithme.

Théorème 2.3 Si n est une puissance de 2, l'algorithme de Karatsuba calcule le produit de deux polynômes de degrés au plus $n-1$ en au plus $9n^{\log_2 3}$ opérations dans $\mathbb{A}$.

Démonstration. Un appel en degré strictement inférieur à n effectue trois appels récursifs en degré strictement inférieur à $n/2$, plus quelques additions, comptées comme suit : l'étape (4) effectue deux additions en taille strictement inférieure à $n/2$; l'étape (6) effectue deux additions en taille strictement inférieure à n ; quant à l'étape (7), A_1 et A_2X^n sont à supports monomiaux distincts, donnant la somme $A_1 + A_2X^n$ sans coût arithmétique, puis la somme totale en une addition en taille strictement inférieure à n. Le coût $K(n)$ satisfait donc à la récurrence

$$K(n) \leq 3K(n/2) + 4n,$$

où le terme $4n$ vient compter le nombre d'additions dans $\mathbb{A}$. Le lemme « diviser pour régner » permet alors de conclure avec $p = q = s = 2$, $m = 3$, $T(n) = 4n$, et $\kappa = 1$ (coût de l'algorithme naïf en degré 0). ■

Le théorème « diviser pour régner » fournit ensuite :

Corollaire 2.4 On peut multiplier deux polynômes de degré n arbitraire en $O(n^{\log_2 3}) = O(n^{1,59})$ opérations dans $\mathbb{A}$.

Une manière d'estimer plus précisément la constante cachée dans cette borne a été présentée au Chapitre 1.

Exercice 2.5 Soit n une puissance de 2. Établir un algorithme hybride de multiplication dans $\mathbb{A}[X]$, qui fait appel à l'algorithme de Karatsuba pour $n > 2^d$ et à l'algorithme naïf pour $n \leq 2^d$. Montrer que la complexité arithmétique $C(n)$ de cet algorithme vérifie $C(n) \leq \gamma(d)n^{\log_2 3} - 8n$ pour tout $n \geq 2^d$, où $\gamma(d)$ est une fonction qui dépend uniquement de d. Trouver la valeur de d qui minimise $\gamma(d)$ et, par comparaison avec le résultat du Théorème 2.3, estimer le gain obtenu par cette optimisation. ■

Exercice 2.6 Est-il possible de multiplier par un algorithme universel, c'est-à-dire indépendant de l'anneau de base $\mathbb{A}$, deux polynômes de degré au plus 1 en utilisant seulement 2 multiplications dans $\mathbb{A}$? ■

Exercice 2.7 — Algorithme de Toom–Cook. Soit $\mathbb{A}$ un anneau et soient A et B deux polynômes de degré au plus 3 dans $\mathbb{A}[X]$.

1. Estimer le nombre de multiplications de $\mathbb{A}$ requises par l'algorithme de Karatsuba pour calculer le produit AB.
2. On suppose que 2, 3 et 5 sont inversibles dans $\mathbb{A}$ et que la division d'un élément de $\mathbb{A}$ par 2, 3 et 5 est gratuite. Donner un algorithme qui multiplie A et B en utilisant au plus 7 multiplications dans $\mathbb{A}$.
3. On suppose que 2, 3 et 5 sont inversibles dans $\mathbb{A}$. Donner un algorithme de multiplication polynomiale dans $\mathbb{A}[X]$ de complexité arithmétique $O(n^{1,4})$.

Dans la suite de l'exercice, on suppose que l'anneau $\mathbb{A}$ est de caractéristique nulle.

4. Montrer que, pour tout entier $\alpha \geq 2$, il existe un algorithme de multiplication polynomiale dans $\mathbb{A}[X]$ de complexité arithmétique $O(n^{\log_\alpha(2\alpha-1)})$.
5. Montrer que pour tout $\varepsilon > 0$, il existe un algorithme de multiplication polynomiale dans $\mathbb{A}[X]$ de complexité arithmétique $O(n^{1+\varepsilon})$, où la constante dans le $O(\cdot)$ dépend de ε, mais pas de n.

■

2.4 Transformée de Fourier rapide

Les méthodes à base de transformée de Fourier rapide (appelée aussi FFT pour *Fast Fourier Transform*) sont ce que l'on sait faire de mieux, à l'heure actuelle, pour multiplier les polynômes. Pour simplifier la présentation, on suppose ici que l'on cherche à multiplier des polynômes F et G dans $\mathbb{A}[X]$, de degrés strictement inférieurs à $n/2$ (ou plus généralement tels que $\deg(FG) < n$).

Idée de l'algorithme

En supposant que l'anneau $\mathbb{A}$ le permette, l'idée générale est synthétisée dans l'Algorithme 2.2. Il s'agit d'évaluer en des points bien choisis, de multiplier les évaluations, et de reconstruire les coefficients du produit à partir de ces valeurs (à condition que cette opération d'interpolation soit possible, voir ci-dessous). Si deux polynômes coïncident sur $1, \omega, \ldots, \omega^{n-1}$, nous verrons que leur différence est un multiple de $X^n - 1$, ce qui justifie la correction de l'algorithme.

Lorsque l'algorithme est employé avec l'hypothèse $\deg H < n$, les coefficients de $H \bmod X^n - 1$ qui sont renvoyés sont bien ceux de H. Le coût des étapes de précalcul et de produit point à point est linéaire en n, et il reste donc à voir comment effectuer rapidement les opérations d'évaluation et d'interpolation.

Entrée F et G deux polynômes, n un entier, et ω une racine principale n-ième de l'unité.
Sortie $H = FG \bmod X^n - 1$.

1. *Précalcul.* Calculer les puissances $\omega^2, \ldots, \omega^{n-1}$.
2. *Évaluation.* Calculer les valeurs :
$$\mathrm{Ev}(F) = \big(F(\omega^0), \ldots, F(\omega^{n-1})\big);\ \mathrm{Ev}(G) = \big(G(\omega^0), \ldots, G(\omega^{n-1})\big).$$
3. *Produit point à point.*
$$\big(\mathrm{Ev}(F), \mathrm{Ev}(G)\big) \mapsto \mathrm{Ev}(FG) = \big(FG(\omega^0), \ldots, FG(\omega^{n-1})\big).$$
4. *Interpolation.*
$$\mathrm{Ev}(FG) \mapsto FG.$$

Algorithme 2.2 – Multiplication de polynômes par transformée de Fourier discrète.

Racines primitives et principales de l'unité

Définition 2.1 L'élément ω de $\mathbb{A}$ est une *racine n-ième de l'unité* si $\omega^n = 1$; c'est une racine n-ième *primitive* de l'unité si de plus $\omega^t \neq 1$ pour $t < n$; c'est une racine n-ième *principale* de l'unité si de plus $\omega^t - 1$ est non diviseur de zéro dans $\mathbb{A}$ pour $t \in \{1, \ldots, n-1\}$ (c'est-à-dire que $\alpha(\omega^t - 1) = 0$ implique $\alpha = 0$).

Exemple 2.1 Si $\mathbb{A}$ est un corps ou même un anneau intègre, les racines principales et primitives coïncident et la condition revient à dire que ω engendre le groupe des racines n-ièmes de l'unité. Par exemple, dans $\mathbb{C}$, -1 n'est pas une racine 4^{e} primitive de l'unité, alors que i l'est. Plus généralement, lorsque $\mathbb{A} = \mathbb{C}$, les racines primitives n-ièmes de l'unité sont de la forme $\exp(2qi\pi/n)$, pour q premier avec n.

Exemple 2.2 Dans $\mathbb{A} = \mathbb{Z}/25\mathbb{Z}$, les premières puissances de 6 sont 6,11,16,21, et 1. Comme par exemple $16 - 1 = 15$ est diviseur de 0, 6 est une racine 5^{e} de l'unité qui est primitive mais n'est pas principale. En revanche, les premières puissances de 7 sont 7,24,18,1, ce qui montre que 7 est une racine principale 4^{e} de 1 dans $\mathbb{A}$.

L'intérêt des racines principales de 1 dans des anneaux qui ne sont même pas intègres apparaît en fin de ce chapitre pour l'algorithme de Schönhage et Strassen. Jusque là, il est suffisant de considérer le cas où $\mathbb{A} = \mathbb{C}$ pour comprendre les idées. Les propriétés que nous utiliserons sont résumées dans le lemme suivant.

Lemme 2.5 Si ω est racine primitive ou principale n-ième de l'unité, alors
1. ω^{-1} aussi ;

2. si $n = pq$ alors ω^p est une racine q-ième de l'unité de même nature que ω ;
3. pour $\ell \in \{1, \dots, n-1\}$, et ω une racine principale de l'unité alors

$$\sum_{j=0}^{n-1} \omega^{\ell j} = 0.$$

Démonstration. Tout d'abord ω est bien inversible : l'identité $\omega^n = 1$ montre que ω^{n-1} est un inverse de ω. Ensuite, ω^{-1} est une racine de l'unité : le produit de l'identité précédente par ω^{-n} donne $1 = \omega^{-n}$. Enfin, elle est principale si ω l'est : lorsque $\omega^t - 1$ n'est pas diviseur de 0, son produit par l'inversible ω^{-t} non plus. Le même argument s'applique dans le cas où ω est primitive.

La deuxième propriété est immédiate. Pour la troisième, le produit de la somme par $1 - \omega^\ell$ télescope les sommants, ce qui donne

$$(1 - \omega^\ell) \sum_{j=0}^{n-1} \omega^{\ell j} = 1 - \omega^{\ell n} = 1 - (\omega^n)^\ell = 0.$$

Comme $1 - \omega^\ell$ n'est pas diviseur de 0, la somme est bien nulle. ■

R Dans la définition de racine primitive ou principale, la condition *t diviseur strict de n* suffit, elle entraîne la propriété pour $t \in \{1, \dots, n-1\}$. En effet, si t n'est pas diviseur de n, son pgcd g avec n l'est. Il existe d'après le théorème de Bézout deux entiers $p, q \in \mathbb{N}$ tels que $g = tp + nq$. Alors, l'égalité $\alpha(\omega^t - 1) = 0$ entraîne $0 = \alpha(\omega^t - 1)(1 + \omega^p + \dots + \omega^{t(p-1)}) = \alpha(\omega^{tp} - 1) = \alpha(\omega^{g-nq} - 1) = \alpha(\omega^g - 1)$, et donc $\alpha = 0$, puisque g est un diviseur strict de n.

Transformée de Fourier rapide

L'opération

$$\mathsf{DFT} : F \in \mathbb{A}[X] \mapsto (F(1), F(\omega), \dots, F(\omega^{n-1})),$$

où ω est une racine principale n-ième de l'unité, s'appelle la *transformée de Fourier discrète*. Son calcul rapide est effectué par un algorithme de type « diviser pour régner ».

Pour appliquer cette idée, supposons que n est pair, $n = 2k$. Alors, $\omega^k = -1$ puisque

$$(\omega^k - 1)(\omega^k + 1) = \omega^n - 1 = 0$$

et le premier facteur n'est pas diviseur de 0. Le polynôme F est décomposé par division euclidienne de deux façons :

$$F = Q_0(X^k - 1) + R_0 \quad \text{et} \quad F = Q_1(X^k + 1) + R_1,$$

avec $\deg R_0 < k$ et $\deg R_1 < k$. Ces décompositions vont nous permettre le calcul de F sur les puissances paires et impaires de ω. En effet, si ℓ est pair, $\omega^{k\ell} = 1$ et donc $F(\omega^\ell) = R_0(\omega^\ell)$. De même, si ℓ est impair, $F(\omega^\ell) = R_1(\omega^\ell)$. Outre l'application

Entrée $F = f_0 + \cdots + f_{n-1}X^{n-1}$; les puissances $1, \omega, \cdots, \omega^{n-1}$ d'une racine n-ième principale de l'unité ω, n étant une puissance de 2.
Sortie $F(1), \ldots, F(\omega^{n-1})$.

1. Si $n = 1$, renvoyer f_0.
2. Sinon, soit $k = n/2$. Calculer

$$R_0(X) = \sum_{j=0}^{k-1} (f_j + f_{j+k})X^j,$$

$$\bar{R}_1(X) = R_1(\omega X) = \sum_{j=0}^{k-1} (f_j - f_{j+k})\omega^j X^j.$$

3. Calculer récursivement $R_0(1), R_0(\omega^2), \ldots, R_0\left((\omega^2)^{k-1}\right)$ et $\bar{R}_1(1), \bar{R}_1(\omega^2), \ldots, \bar{R}_1\left((\omega^2)^{k-1}\right)$.
4. Renvoyer $R_0(1), \bar{R}_1(1), R_0(\omega^2), \bar{R}_1(\omega^2), \ldots, R_0\left((\omega^2)^{k-1}\right), \bar{R}_1\left((\omega^2)^{k-1}\right)$.

Algorithme 2.3 – Transformée de Fourier rapide (FFT).

récursive, le point crucial qui est la source de l'efficacité de l'algorithme de transformée de Fourier rapide (Algorithme 2.3) et qui conduit au choix de racines primitives de l'unité, est que le calcul de R_0 et R_1 est très simple (étape (2)). Lors des appels récursifs, les puissances de ω qui sont utilisées sont des ω^{2^i}, qui sont bien des racines primitives d'après le Lemme 2.5.

Théorème 2.6 L'Algorithme 2.3 de transformée de Fourier rapide requiert au plus $\frac{3n}{2}\log n$ opérations dans $\mathbb{A}$; les multiplications font toutes intervenir une puissance de ω.

Démonstration. Puisque les puissances de ω sont connues, le coût de l'appel en degré n est d'au plus $2 \times n/2$ additions et soustractions (pour le calcul de R_0 et R_1) et de $n/2$ multiplications (pour le calcul de $\bar{R}_1$), plus deux appels récursifs en degré $n/2$. Sa complexité $F(n)$ satisfait donc à la récurrence :

$$F(n) \leq \frac{3n}{2} + 2F\left(\frac{n}{2}\right)$$

et le lemme « diviser pour régner » (avec $p = q = m = s = 2$, $T(n) = 3n/2$ et $\kappa = 0$) permet de conclure. ■

Exercice 2.8 Montrer que l'Algorithme 2.3 requiert $n \log n$ additions dans $\mathbb{A}$, $\frac{1}{2}n\log n$ multiplications d'éléments de $\mathbb{A}$ par des puissances de ω, mais aucune autre multipli-

cation dans $\mathbb{A}$. ∎

R La transformée de Fourier discrète est un morphisme d'algèbres sur $\mathbb{A}$ de $\mathbb{A}[X]/(X^n-1)$ dans $\mathbb{A}^n$ avec comme multiplication dans $\mathbb{A}^n$ la multiplication coordonnée par coordonnée. Cette observation permet d'économiser des transformées inverses en effectuant plusieurs calculs directement sur les transformées. Une application typique de cette observation est le produit scalaire, ou plus généralement le produit de matrices.

Interpolation

En termes matriciels, l'opération $F \mapsto \mathrm{Ev}(F)$ est linéaire et sa matrice (pour des polynômes F de degré au plus $n-1$, dans la base monomiale $\{1, X, \ldots, X^{n-1}\}$) est la matrice de Vandermonde

$$V_\omega = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \cdots & \omega^{n-1} \\ \vdots & & & \vdots \\ 1 & \omega^{n-1} & \cdots & \omega^{(n-1)^2} \end{pmatrix}.$$

Lemme 2.7 Si $\omega \in \mathbb{A}$ est racine n-ième principale de l'unité, alors $V_{\omega^{-1}} V_\omega = nI_n$.

Démonstration. D'après le Lemme 2.5, ω est inversible. La matrice $V_{\omega^{-1}}$ est donc bien définie. Le coefficient de la i-ième ligne et j-ième colonne du produit s'écrit

$$\sum_{k=0}^{n-1} \omega^{(i-1)k} \omega^{-(j-1)k} = \sum_{k=0}^{n-1} \omega^{(i-j)k}.$$

D'après le Lemme 2.5, cette dernière somme est nulle pour $0 < i-j < n$. Par symétrie des matrices, elle est aussi nulle si $0 < j-i < n$. Les coefficients en dehors de la diagonale sont donc tous nuls. Sur la diagonale, la somme comporte n termes, tous égaux à 1, d'où le résultat. ∎

Autrement dit, l'interpolation sur les puissances de ω est calculée efficacement en la ramenant à une FFT sur les puissances de ω^{-1}, qui est bien principale d'après le Lemme 2.5.

Conclusion

Les résultats de cette section sont résumés dans le théorème suivant.

Théorème 2.8 Si 2 est inversible dans $\mathbb{A}$ et n une puissance de 2, étant donnée une racine principale n-ième dans $\mathbb{A}$, le produit de deux polynômes dont la somme des degrés est inférieure à n peut être calculé en $\frac{9}{2} n \log n + O(n)$ opérations dans $\mathbb{A}$. Seuls n des produits sont entre deux éléments de $\mathbb{A}$ qui ne sont pas des puissances de ω.

Démonstration. La complexité de l'Algorithme 2.2 est de 3 FFT en degré n, soit $\frac{9}{2}n\log n$ opérations, plus $O(n)$ divisions par n pour l'interpolation (ce qui est possible puisque 2 est inversible) et $O(n)$ multiplications pour calculer les puissances de ω. Les multiplications de la FFT font intervenir des puissances de ω, les autres sont celles de l'étape de produit point à point, au nombre de n. ■

Exercice 2.9 Montrer que sous les hypothèses du théorème précédent, on peut multiplier ces polynômes en utilisant $3n\log n + O(n)$ additions dans $\mathbb{A}$, plus $\frac{3}{2}n\log n + O(n)$ multiplications par des puissances de ω, n multiplications arbitraires dans $\mathbb{A}$, et n divisions par n. ■

Exercice 2.10 Soit n dans $\mathbb{N}$, soit n_0 la plus petite puissance de 2 supérieure ou égale à n, et supposons qu'il existe dans $\mathbb{A}$ une racine n_0-ième principale de l'unité. Cette racine étant connue, on peut multiplier les polynômes de degré au plus $\frac{1}{2}n-1$ en $9n\log n + O(n)$ opérations dans $\mathbb{A}$. ■

Exercice 2.11 Soit n une puissance de 2, et supposons qu'on dispose d'une racine n-ième principale de l'unité $\omega \in \mathbb{A}$. Soit P et Q deux polynômes dans $\mathbb{A}[X]$ de degré au plus $n-1$. Supposons que les coefficients de $X^0, X^1, \ldots, X^{n-1}$ du produit $R = PQ$ sont connus. Montrer que R peut être calculé en $\frac{9}{2}n\log n + O(n)$ opérations dans $\mathbb{A}$. ■

Mais où sont nos racines ?

Même dans le cas favorable où $\mathbb{A}$ est un corps, il n'y existe pas nécessairement toutes les racines primitives de l'unité nécessaires pour appliquer les techniques des algorithmes précédents. Dans le cas particulier des corps finis, on sait donner une réponse précise à cette question d'existence.

Proposition 2.9 Le corps fini à q éléments $\mathbb{F}_q$ contient une racine n-ième primitive de l'unité si et seulement si n divise $q-1$.

Exercice 2.12
1. Prouver la proposition précédente.
2. Montrer que si n divise $q-1$ et si α est un élément primitif de $\mathbb{F}_q$ (c'est-à-dire tel que α engendre le groupe multiplicatif $(\mathbb{F}_q \setminus \{0\}, \times)$) alors $\alpha^{(q-1)/n}$ est une racine n-ième primitive de l'unité.

■

Pour pouvoir appliquer le Théorème 2.6 avec n assez grand et $\mathbb{A} = \mathbb{Z}/p\mathbb{Z}$ (c'est-à-dire $q = p$ premier), ce résultat mène à la notion de *premiers de Fourier*, qui sont les nombres premiers p tels que $p-1$ soit divisible par une grande puissance de 2, donc de la forme $\ell 2^e + 1$, avec e « suffisamment grand » (qui est appelé l'*exposant* de p). Par exemple,

$$4179340454199820289 = 29 \times 2^{57} + 1$$

est un tel nombre premier. Ainsi, dans $\mathbb{Z}/4179340454199820289\mathbb{Z}$, on dispose de racines primitives 2^{57}-ième de l'unité (21 en est une) ; on peut donc y multiplier des polynômes de degrés colossaux par cet algorithme en $O(n\log n)$ opérations.

Ces premiers sont donc très utiles pour la FFT sur des corps finis. On peut montrer qu'il y a approximativement $(x/\log x)/2^{e-1}$ premiers de Fourier d'exposant e inférieurs à x (à rapprocher du théorème des nombres premiers qui montre l'existence d'approximativement $x/\log x$ premiers inférieurs à x). Il s'ensuit qu'il y a environ 130 corps finis de la forme $k = \mathbb{F}_p$ tel que p a la taille d'un (demi-)mot machine (32 bits) et tel que dans k on puisse multiplier par FFT des polynômes de degré de l'ordre d'un million. Les racines de l'unité de $\mathbb{F}_p$ peuvent être calculées à partir des éléments primitifs. Si on choisit au hasard un élément de $\mathbb{F}_p$, lorsque $p \to \infty$, la probabilité qu'il soit primitif tend vers $6/\pi^2$, soit plus de $0,6$.

2.5 L'algorithme de Schönhage–Strassen

Quand les racines de l'unité font défaut, il reste possible de faire fonctionner les idées à base de transformée de Fourier. Ceci est réalisé par l'algorithme de Schönhage et Strassen. Cet algorithme s'applique quel que soit l'anneau de base, pourvu que 2 y soit inversible ; l'idée est de rajouter les racines de l'unité qui manquent en étendant l'anneau de base de manière judicieuse.

Racines virtuelles de l'unité

Le point de départ de l'algorithme est résumé dans le résultat suivant.

Lemme 2.10 Si 2 est inversible dans $\mathbb{A}$ et n est une puissance de 2, alors $\omega = X \bmod (X^n + 1)$ est une racine $2n$-ième principale de l'unité dans $\mathbb{A}[X]/(X^n + 1)$.

Démonstration. Comme $\omega^n = -1$, ω est bien une racine $2n$-ième de l'unité. Il reste à prouver qu'elle est principale. D'après la remarque en page 42, il s'agit de montrer que $\omega^t - 1$ n'est pas diviseur de 0, quel que soit le diviseur strict t de $2n$. Puisque n est une puissance de 2, l'entier t divise n, et donc $\omega^t - 1$ divise l'élément $\omega^n - 1 = -2$ qui est supposé inversible, ce qui conclut la preuve. ■

Exemple 2.3 Même lorsque $\mathbb{A}$ est un corps, ces anneaux ne sont en général pas intègres. Par exemple, dans $\mathbb{Z}/3\mathbb{Z}$, la factorisation $Y^4 + 1 = (Y^2 + Y + 2)(Y^2 + 2Y + 2)$ exhibe deux polynômes diviseurs de 0 dans $\mathbb{Z}/3\mathbb{Z}[Y]/(Y^4 + 1)$.

L'algorithme

La méthode est résumée dans l'Algorithme 2.4, qui calcule le produit FG modulo $X^n + 1$: ceci permet une utilisation récursive mais ajoute une petite complication pour le produit par FFT, qui calcule un produit modulo $X^n + 1$, d'où les changements de variables et choix de racines de l'unité dans l'étape 2.

Exemple 2.4 Un exemple jouet permet de mieux comprendre comment se déroule cet algorithme et pourquoi il est correct. La multiplication de polynômes de $\mathbb{Z}/3\mathbb{Z}[X]$ par FFT ne peut pas aller au-delà du degré 2, puisqu'il n'y a pas de racine

Entrée F et G de degrés strictement inférieurs à n, où $n = 2^k$, $k > 2$.
Sortie FG mod $X^n + 1$.

1. Soient $d = 2^{\lfloor k/2 \rfloor}$ et $\delta = n/d$. Récrire F et G sous la forme
$$\bar{F}(X, Y) = F_0(X) + F_1(X)Y + \cdots + F_{\delta-1}(X)Y^{\delta-1},$$
$$\bar{G}(X, Y) = G_0(X) + G_1(X)Y + \cdots + G_{\delta-1}(X)Y^{\delta-1},$$
de sorte que les F_i et G_i aient des degrés strictement inférieurs à d et que $F(X) = \bar{F}(X, X^d)$ et $G(X) = \bar{G}(X, X^d)$.
2. Calculer $\bar{H} := \bar{F}\bar{G} \bmod Y^\delta + 1$ dans $\mathbb{B}[Y]$ par FFT, où $\mathbb{B} = \mathbb{A}[X]/(X^{2d} + 1)$ et les produits dans $\mathbb{B}$ sont effectués récursivement. Pour cela,
   — si k est pair, calculer $\bar{H}(X, X^2Y) \bmod Y^\delta - 1$ avec X^4 comme racine de l'unité, puis reconstruire $\bar{H}$ avec $X^{-2} = -X^{2d-2}$;
   — si k est impair, calculer $\bar{H}(X, XY) \bmod Y^\delta - 1$ avec X^2 comme racine de l'unité, puis reconstruire $\bar{H}$ avec $X^{-1} = -X^{2d-1}$.
3. Renvoyer $H(X, X^d)$.

Algorithme 2.4 – Algorithme de Schönhage–Strassen sur les polynômes.

n-ième de l'unité d'ordre plus grand. Pour calculer le produit de

$$F = 1 + X^7 \quad \text{et} \quad G = 1 + X + X^6 \quad \text{modulo } X^{16} + 1$$

dans $\mathbb{Z}/3\mathbb{Z}[X]$, l'algorithme introduit $d = 4, \delta = 4$ et

$$\bar{F} = 1 + X^3Y, \quad \bar{G} = 1 + X + X^2Y,$$

et cherche à calculer le produit de ces deux polynômes

$$\bar{H}(X, Y) = \bar{F}(X, Y)\bar{G}(X, Y) \bmod Y^4 + 1,$$

vus comme polynômes en Y à coefficients dans $\mathbb{Z}/3\mathbb{Z}[X]/(X^8 + 1)$. Ce sera suffisant pour connaître le résultat puisqu'en $Y = X^4$, on a $Y^4 + 1 = X^{16} + 1$, polynôme modulo lequel l'algorithme calcule le produit.

Pour se ramener à la situation d'une FFT, il suffit de considérer $\bar{H}(X, X^2Y)$: le remplacement de Y par X^2Y dans l'identité ci-dessus montre qu'on calcule alors modulo $X^8Y^4 + 1 = -Y^4 + 1$. On commence donc par calculer une DFT de

$$\bar{F}(X, X^2Y) = 1 + X^5Y, \quad \bar{G}(X, X^2Y) = 1 + X + X^4Y$$

avec X^4 comme racine 4e de l'unité. On obtient les évaluations en 1, X^4, $X^8 = -1$, $X^{12} = -X^4$, à savoir $(1 + X^5, 1 - X, 1 - X^5, 1 + X)$ et $(1 + X + X^4, X, 1 + X - X^4, 2 + X)$ qui se calculent par une FFT très peu chère, puisque les multiplications par des puissances de X ne demandent pas d'opération arithmétique.

Ensuite, il faut multiplier deux à deux ces valeurs dans $\mathbb{Z}/3\mathbb{Z}[X]/(X^8 + 1)$ pour obtenir les évaluations de $\bar{H}$. L'algorithme procède récursivement parce que c'est encore possible (lorsque n devient trop petit, $2d = n$ ce qui mènerait à une boucle infinie ; on a alors recours à une autre méthode, par exemple un produit naïf en ces petits degrés.) Ces calculs récursifs mènent aux valeurs suivantes pour $\bar{H}(X, X^2Y)$ en 1, X^4, $X^8 = -1$, $X^{12} = -X^4$:

$$c_0 = 1 + X^4 + X^5 + X^6, c_1 = X - X^2, c_2 = 1 - X^4 - X^5 - X^6, c_3 = 2 + X^2.$$

L'interpolation par FFT consiste à évaluer le polynôme $c_0 + c_1Y + c_2Y^2 + c_3Y^3$ en les puissances de $X^{-4} = -X^4$, à savoir 1, $-X^4$, -1, X^4, ce qui donne

$$1 + X, X^4 + X^5 + X^6, 2X, 0$$

et donc finalement (après multiplication par $1/n = 1/4 = 1$)

$$\bar{H}(X, X^2Y) = 1 + X + X^4(1 + X + X^2)Y + 2XY^2.$$

En utilisant $X^{-2} = -X^6$ pour remplacer Y par YX^{-2}, il vient

$$\bar{H}(X, Y) = 1 + X + X^2(1 + X + X^2)Y + X^5Y^2.$$

Le résultat s'obtient enfin en évaluant en $Y = X^4$:

$$FG = 1 + X + X^6(1 + X + X^2) + X^{13} + X^{12} = 1 + X + X^6 + X^7 + X^8 + X^{13}.$$

En ce qui concerne la complexité, la première étape ne demande pas d'opération arithmétique, et l'évaluation de la dernière étape se fait en complexité linéaire, ainsi que les changements de Y en X^kY avant et après la FFT. Ensuite, d'après le Théorème 2.8, la multiplication $\bar{F}\bar{G}$ nécessite :

- $O(d \log d)$ opérations $(+, -)$ dans $\mathbb{B}$, chacune en $O(d)$ opérations dans $\mathbb{A}$;
- $O(d \log d)$ multiplications par une puissance de ω dans $\mathbb{B}$, chacune en $O(d)$ opérations dans $\mathbb{A}$, par simples décalages des indices et éventuels changements de signes ;
- $O(d)$ divisions par d dans $\mathbb{B}$, chacune en $O(d)$ opérations dans $\mathbb{A}$;
- au plus δ produits dans $\mathbb{B}$.

Le coût $C(n)$ de cet algorithme obéit donc à l'inégalité

$$C(n) \leq Kd \log d \times d + \delta C(2d)$$

où K est une constante indépendante de n et de $\mathbb{A}$. En divisant par $n = d\delta$, cette inégalité devient

$$\frac{C(n)}{n} \leq K\frac{d}{\delta} \log d + 2\frac{C(2d)}{2d} \leq K' \log n + 2\frac{C(2d)}{2d}$$

pour n assez grand, et une constante K′ indépendante de n et de $\mathbb{A}$. Ici $2d = 2^{\lfloor k/2 \rfloor + 1}$, et pour se ramener au théorème diviser pour régner, il sera plus commode de faire apparaître $\lfloor (k+1)/2 \rfloor = \lceil k/2 \rceil$. Pour cela, on considère donc d'abord

$$\frac{\mathrm{C}(2n)}{2n} \leq \mathrm{K}' \log n + 2 \frac{\mathrm{C}(2^{\lceil k/2 \rceil + 1})}{2^{\lceil k/2 \rceil + 1}}$$

et on pose $c(k) = \mathrm{C}(2^{k+1})/2^{k+1}$, ce qui donne par récriture

$$c(k) \leq \mathrm{K}'' k + 2c(\lceil k/2 \rceil),$$

pour une nouvelle constante K″. Le résultat final s'en déduit par application du théorème « diviser pour régner ».

Théorème 2.11 Soit $\mathbb{A}$ un anneau dans lequel 2 est inversible (d'inverse connu). Alors, deux polynômes de $\mathbb{A}[\mathrm{X}]$ de degrés au plus n peuvent être multipliés en $\mathrm{O}(n \log n \log\log n)$ opérations $(+, -, \times)$ dans $\mathbb{A}$.

Il est possible d'étendre cette idée au cas où 3 est inversible (FFT triadique), et plus généralement à un anneau quelconque. On obtient alors l'algorithme de complexité $\mathrm{O}(n \log n \log\log n)$ mentionné dans l'introduction.

2.6 Algorithmes pour les entiers

Les algorithmes ainsi que les résultats présentés ci-dessus s'étendent à la multiplication des entiers [1]. Nous allons brièvement présenter cette problématique à travers un exemple.

Soient à multiplier les entiers 2087271 et 1721967, qu'on suppose donnés en base 2,

$$\mathrm{A} = 11111101100101100111 \quad \text{et} \quad \mathrm{B} = 11010010001100110111,$$

chacun ayant D = 21 chiffres binaires. On peut ramener leur multiplication à un produit de polynômes. Plus exactement, on associe à A et B les polynômes de degré strictement inférieur à D :

$$\mathrm{P} = \mathrm{X}^{20} + \mathrm{X}^{19} + \mathrm{X}^{18} + \mathrm{X}^{17} + \mathrm{X}^{16} + \mathrm{X}^{15} + \mathrm{X}^{14} + \mathrm{X}^{12} + \mathrm{X}^{11} + \mathrm{X}^{8} + \mathrm{X}^{6} + \mathrm{X}^{5} + \mathrm{X}^{2} + \mathrm{X} + 1$$

et

$$\mathrm{Q} = \mathrm{X}^{20} + \mathrm{X}^{19} + \mathrm{X}^{17} + \mathrm{X}^{14} + \mathrm{X}^{10} + \mathrm{X}^{9} + \mathrm{X}^{6} + \mathrm{X}^{5} + \mathrm{X}^{3} + \mathrm{X}^{2} + \mathrm{X} + 1.$$

La stratégie est la suivante : on calcule R = PQ dans $\mathbb{Z}[\mathrm{X}]$ puis on évalue R en X = 2. Pour multiplier P et Q dans $\mathbb{Z}[\mathrm{X}]$, il suffit d'effectuer leur produit dans $\mathbb{A}[\mathrm{X}] = (\mathbb{Z}/p\mathbb{Z})[\mathrm{X}]$, où p est un nombre premier tel que $2\mathrm{D} > p > \mathrm{D}$. Par le Théorème 2.11, cette multiplication polynomiale en degré D peut se faire en $\mathrm{O}(\mathrm{D} \log \mathrm{D} \log\log \mathrm{D})$ opérations $(+, -, \times)$ dans $\mathbb{A} = \mathbb{Z}/p\mathbb{Z}$.

1. Historiquement, les algorithmes pour la multiplication des entiers ont été introduits *avant* leurs homologues polynomiaux, alors que ces derniers sont souvent bien plus simples à énoncer.

Puisque chaque opération de $\mathbb{A}$ nécessite au plus $O(\log^2 D)$ opérations binaires, il s'ensuit que le coût binaire du calcul de R est de $O(D \log^3 D \log\log D)$. Ici $p = 23$ et R (écrit en base 2) vaut $R = X^{40} + 10X^{39} + 10X^{38} + 11X^{37} + 11X^{36} + 11X^{35} + 100X^{34} + 11X^{33} + 11X^{32} + 100X^{31} + 11X^{30} + 100X^{29} + 101X^{28} + 11X^{27} + 101X^{26} + 1000X^{25} + 101X^{24} + 101X^{23} + 1000X^{22} + 1001X^{21} + 1010X^{20} + 1000X^{19} + 111X^{18} + 1000X^{17} + 110X^{16} + 110X^{15} + 110X^{14} + 11X^{13} + 100X^{12} + 110X^{11} + 100X^{10} + 11X^9 + 100X^8 + 100X^7 + 100X^6 + 11X^5 + 10X^4 + 11X^3 + 11X^2 + 10X + 1$. Enfin, l'évaluation de R en 2 revient à gérer les retenues et cela peut se faire en un coût négligeable. Ici $AB = R(2) = 11010001001101011110110111011011010101001$, ou encore, en écriture décimale $AB = 3594211782057$.

Une légère amélioration est possible si l'on choisit pour p un premier de Fourier supérieur à 2D. Dans notre exemple, on peut prendre $p = 193 = 3 \cdot 2^6 + 1$. En effet, il existe dans $\mathbb{A} = \mathbb{Z}/193\mathbb{Z}$ une racine primitive $\omega = 125$, d'ordre $2^6 = 64$, donc supérieur à $2D = 40$. Ce choix mène à un algorithme de complexité binaire $O(D \log^3 D)$.

Une autre approche est de calculer PQ dans $\mathbb{Z}[X]$ par DFT en calculant avec des nombres complexes représentés par des approximations numériques flottantes. Une estimation des erreurs numériques montre qu'il suffit de calculer en précision fixe $O(\log^2 D)$. La complexité binaire est donc toujours $O(D \log^3 D)$ via cette approche.

On peut faire un peu mieux, en remplaçant la base 2 par une base B telle que $B \log B$ soit de l'ordre de $\log D$ et en appliquant l'un des raisonnements précédents ; on peut aboutir ainsi à un algorithme de complexité binaire $O(D \log^2 D)$. Qui plus est, en appelant récursivement l'algorithme pour multiplier les petits entiers, on peut descendre cette complexité à $O(D \log D \log\log D \log\log\log D \ldots)$.

Une meilleure solution est cependant détenue par la version entière, à base de FFT dans des anneaux de type $\mathbb{Z}/(2^{2^k} + 1)\mathbb{Z}$, de l'algorithme de Schönhage–Strassen, dont nous nous contentons de mentionner l'existence.

Théorème 2.12 On peut multiplier deux entiers de D chiffres binaires en utilisant $O(D \log D \log\log D)$ opérations binaires.

En développant largement ces techniques de FFT, la complexité du produit entier peut être réduite à $O\left(n \log n \cdot 2^{O(\log^* n)}\right)$, où $\log^* n$ est le nombre d'itérations du logarithme pour ramener n en dessous de 1. Par exemple (pour la base 2), $\log^* 256 = 4$, car $\log 256 = 8$, $\log 8 = 3$, $1 < \log 3 < 2$ et enfin $\log\log 3 < 1$.

2.7 Un concept important : les fonctions de multiplication

Un bon nombre des résultats de complexité donnés dans la suite de cet ouvrage reposent sur la notion de *fonction de multiplication*. Une fonction $M : \mathbb{N} \to \mathbb{N}$ sera dite fonction de multiplication polynomiale (pour un anneau $\mathbb{A}$) si :

— on peut multiplier les polynômes de $\mathbb{A}[X]$ de degrés au plus n en au plus $M(n)$ opérations dans $\mathbb{A}$;

— M vérifie l'inégalité $M(n + n') \geq M(n) + M(n')$.

Ainsi, on sait d'après ce qui précède que des fonctions de la forme $n^{\log_2 3}$ ou $n \log n$ sont (à des constantes près) des fonctions de multiplication (respectivement pour tous les anneaux ou ceux qui permettent la transformée de Fourier). L'intérêt de cette

notion est qu'elle permet d'énoncer des résultats de complexité indépendants du choix de l'algorithme utilisé pour multiplier les polynômes (même si parfois cela mène à des estimations légèrement pessimistes, parce que les algorithmes n'exploitent pas la remarque page 44).

La seconde condition est utilisée pour écarter l'hypothèse d'une fonction qui croît trop lentement (si M est constante, elle ne vérifie pas cette condition). Concrètement, elle permettra par exemple d'établir que dans des algorithmes du type « inversion de série formelle par l'opérateur de Newton » (Chapitre 3), le coût total est essentiellement celui de la dernière étape.

De la même manière, on est amené à introduire la notion de *fonction de multiplication* pour les entiers. Une fonction $\mathsf{M}_{\mathbb{Z}} : \mathbb{N} \to \mathbb{N}$ est une fonction de multiplication pour les entiers si :

— on peut multiplier des entiers de n chiffres en $\mathsf{M}_{\mathbb{Z}}(n)$ opérations binaires ;
— $\mathsf{M}_{\mathbb{Z}}$ vérifie l'inégalité $\mathsf{M}_{\mathbb{Z}}(n+n') \geq \mathsf{M}_{\mathbb{Z}}(n) + \mathsf{M}_{\mathbb{Z}}(n')$.

Ce concept est très proche de son analogue polynomial et les contextes d'utilisation sont souvent les mêmes : on utilise la seconde condition pour contrôler les coûts de multiplication lors de l'analyse d'algorithmes récursifs.

Exercices

Exercice 2.13 Soit $\mathbb{A}$ un anneau, soit $a \in \mathbb{A}$ et soit P un polynôme de $\mathbb{A}[X]$, de degré au plus n. On se propose de calculer le polynôme $Q(X) = P(X+a)$.

1. Donner un algorithme direct pour le calcul de Q et estimer sa complexité en termes de n ;
2. Montrer qu'il est possible de calculer Q(X) en utilisant $O\big(\mathsf{M}(n)\log n\big)$ opérations $(+,-,\times)$ dans $\mathbb{A}$. (Un algorithme en $O(\mathsf{M}(n))$ opérations sera donné au Chapitre 3 par somme composée, et une autre solution en $O(\mathsf{M}(n))$ est donnée en Exercice 33.2 en Annexe.)

■

Exercice 2.14 Soit $\mathbb{A}$ un anneau, soient $\kappa, n \in \mathbb{N}$ et soit P un polynôme de $\mathbb{A}[X]$, de degré au plus n.

1. Donner un algorithme direct pour le calcul de $P(X)^\kappa$ et estimer sa complexité en fonction de κ et de n ;
2. Montrer qu'il est possible de calculer $P(X)^\kappa$ en n'utilisant que $O\big(\mathsf{M}(n\kappa)\big)$ opérations $(+,-,\times)$ dans $\mathbb{A}$.

■

Exercice 2.15 — Produit court. Pour deux polynômes $A, B \in \mathbb{Q}[X]$ de degré strictement inférieur à n, le polynôme AB mod X^n est appelé *produit court* de A et B. Le but de cet exercice est de montrer qu'il est possible de calculer plus efficacement le produit court que le produit entier AB. Le gain est d'un facteur constant.

1. Prouver que tel est le cas avec la multiplication polynomiale naïve.

On suppose que l'on dispose d'un algorithme de multiplication polynomiale qui, en degré n, utilise $\mathsf{M}_\alpha(n) = Kn^\alpha$ opérations dans $\mathbb{Q}$, avec $\alpha > 1$ et $K > 0$.

2. Donner un algorithme de type « diviser pour régner » pour le calcul du produit court de A, B.
3. Montrer que la complexité de cet algorithme est $\frac{1}{2^\alpha-2}\mathsf{M}_\alpha(n) + \mathrm{O}(n\log n)$. En déduire l'impact sur la multiplication naïve, et celle de Karatsuba.
4. Modifier la conception de l'algorithme précédent, en découpant en deux sous-problèmes de tailles βn et $(1-\beta)n$, avec $1/2 \le \beta \le 1$ laissé en paramètre. Prouver que le nouvel algorithme utilise

$$\frac{\beta^\alpha}{1-2(1-\beta)^\alpha}\cdot \mathsf{M}_\alpha(n) + \mathrm{O}(n\log n)$$

   opérations dans $\mathbb{Q}$.
5. Montrer que le minimum de la fonction $\beta \mapsto \frac{\beta^\alpha}{1-2(1-\beta)^\alpha}$ est atteint en $\beta_{\min} = 1 - \left(\frac{1}{2}\right)^{\frac{1}{\alpha-1}}$ et en déduire qu'on peut calculer le produit court en $\mathrm{C}(\alpha)\mathsf{M}_\alpha(n) + \mathrm{O}(n\log n)$ opérations dans $\mathbb{Q}$, où

$$\mathrm{C}(\alpha) = \frac{\left(2^{\frac{1}{\alpha-1}}-1\right)^\alpha}{2^{\frac{\alpha}{\alpha-1}}-2}.$$

6. Conclure sur le gain obtenu (à l'aide d'un calcul numérique).

■

Exercice 2.16 Soient $\mathrm{P}_1, \ldots, \mathrm{P}_t$ des polynômes de $\mathbb{A}[\mathrm{X}]$, de degrés $d_1, \ldots, d_t$. Montrer que le produit $\mathrm{P}_1 \cdots \mathrm{P}_t$ peut s'effectuer en $\mathrm{O}(\mathsf{M}(n)\log t)$ opérations dans $\mathbb{A}$, où $n = \sum_i d_i$.

■

Notes

Le mathématicien russe A. N. Kolmogorov avait conjecturé au début des années 1960 qu'il serait impossible de multiplier deux entiers en un coût binaire sous-quadratique. En 1962 cette conjecture fut infirmée par Karatsuba et Ofman [KO63]. Une généralisation de leur algorithme a été proposée peu de temps après par Toom et Cook [Coo66 ; Too63]. L'algorithme de Toom–Cook a une complexité binaire de $\mathrm{O}(n \cdot 32^{\sqrt{\log_2 n}})$ pour multiplier deux entiers de taille binaire n ; ce résultat peut être importé dans le monde polynomial (voir l'Exercice 2.7 pour une version plus faible). L'algorithme de FFT a une longue histoire, qui remonte à Gauss [Coo90 ; CT93 ; HJB85]. Il s'agit d'un progrès algorithmique très célèbre : Dongarra et Sullivan [DS00] le placent parmi les dix algorithmes qui ont le plus marqué le développement des sciences de l'ingénieur du 20e siècle. L'article fondateur de Cooley et Tukey [CT65] est l'un des plus cités en informatique [2]. La variante de la DFT décrite dans ce chapitre, appelée *decimation-in-frequency* en anglais, est due à Gentleman et Sande [GS66] ; il s'agit d'une version duale (au sens du principe de transposition de Tellegen évoqué au Chapitre 12) de l'algorithme *decimation-in-time* [CT65]. De nombreux livres [Bri74 ; Nus81 ; Van92] et articles [Bera ; DV90 ; FJ05] permettent de se familiariser avec la

2. Plus de 2000 citations, d'après la base bibliographique *Science Citation Index (SCI ®)*.

myriade de techniques de type FFT ; en particulier, Frigo et Johnson [FJ05] décrivent l'une des implantations les plus efficaces de la transformée de Fourier complexe (appelée FFTW, pour *Fastest Fourier Transform in the West*.)

Une avancée importante a été la découverte de Schönhage et Strassen [SS71] du résultat équivalent pour la multiplication des nombres entiers. Pendant longtemps, on a cru que cet algorithme ne pourrait présenter qu'un intérêt purement théorique.

À ce jour, la plupart des logiciels généralistes de calcul formel disposent d'une multiplication rapide d'entiers qui inclut la FFT. Maple, Mathematica et Sage utilisent la bibliothèque GMP. Il faut noter que cette bibliothèque est le résultat d'un travail de nombreuses années, qui comporte une partie importante de code assembleur consacré à la multiplication sur chacun des processeurs produits dans une période récente.

L'analogue polynomial de l'algorithme de Schönhage–Strassen traité en Section 2.5 a été suggéré par Nussbaumer [Nus80]. Le cas particulier de la caractéristique 2 est traité par Schönhage [Sch77]. Plus récemment, Cantor et Kaltofen [CK91] ont donné un algorithme qui multiplie des polynômes de degré n sur une algèbre $\mathbb{A}$ (non nécessairement commutative, ni associative) en $O(n \log n \log\log n)$ opérations dans $\mathbb{A}$.

En ce qui concerne la terminologie, nous suivons Demazure [Dem97] pour la définition des racines principales de l'unité. D'autres auteurs [Für09] utilisent la troisième propriété du Lemme 2.5. D'autres enfin [GG03] les appellent simplement primitives.

La borne de complexité binaire $O\left(n \log n \cdot 2^{O(\log^* n)}\right)$ mentionnée en fin de Section 2.6, pour multiplier des entiers de n chiffres, a été obtenue récemment par Fürer [De+08 ; Für07 ; Für09]. Harvey, van der Hoeven et Lecerf ont ensuite montré que l'algorithme de Fürer pouvait être un peu modifié pour atteindre un coût binaire en $O\left(n \log n \cdot 16^{\log_2^* n}\right)$ [HHL16]. Puis ils ont conçu un autre algorithme ayant un coût $O\left(n \log n \cdot 8^{\log_2^* n}\right)$. À ce jour, on ne connaît pas de borne analogue pour la complexité arithmétique de la multiplication polynomiale en degré n en toute généralité. Néanmoins si les coefficients sont dans le corps fini $\mathbb{F}_q$ alors la complexité binaire du produit en degré n est en $O\left(n \log q \log(n \log q) \cdot 8^{\log_2^*(n \log q)}\right)$ [HHL14].

Un problème ouvert est de trouver un algorithme de multiplication polynomiale en complexité $O(n)$, ou de prouver qu'un tel algorithme n'existe pas. Morgenstern [Mor73] a donné une première réponse partielle, en montrant que dans un modèle simplifié où $\mathbb{A} = \mathbb{C}$ et où les seules opérations admises sont les additions et les multiplications par des nombres complexes de module inférieur à 1, au moins $\frac{1}{2} n \log n$ opérations sont *nécessaires* pour calculer une DFT en taille n. Plus récemment, Bürgisser et Lotz [BL04] ont étendu ce type de borne à la multiplication dans $\mathbb{C}[X]$. Concernant les entiers, Cook et Aanderaa [CA69] ont prouvé une borne inférieure de la forme $cn \log n/(\log\log n)^2$ pour la multiplication en taille binaire n sur une machine de Turing. Cette borne a été améliorée à $\Omega(n \log n)$ pour un modèle restreint (*multiplication en ligne*) [PFM74].

Malgré la simplicité de leur idée de base, les algorithmes de Karatsuba et de Toom–Cook soulèvent encore des questions délicates ; plusieurs travaux récents traitent diverses généralisations (degrés déséquilibrés ou non-puissances, caractéristique non nulle de l'anneau des scalaires) [BZ07 ; CH07 ; Mon05 ; WP06]. Par exemple, connaître le nombre minimum de multiplications $\mathsf{sm}(n)$ (resp. $\mathsf{sm}_p(n)$) suffisantes pour multi-

plier des polynômes de degré donné n sur un anneau arbitraire (resp. sur un anneau de caractéristique p) est un problème important et difficile. Pour les premières valeurs de n, les meilleures bornes connues actuellement, dues à Montgomery [Mon05], sont $\mathsf{sm}(1) \leq 3$, $\mathsf{sm}(2) \leq 6$, $\mathsf{sm}(3) \leq 9$, $\mathsf{sm}(4) \leq 13$, $\mathsf{sm}(5) \leq 17$ et $\mathsf{sm}(6) \leq 22$. L'optimalité de ces bornes est un problème difficile. Pour $\mathsf{sm}_p(n)$, on sait par exemple que $\mathsf{sm}_2(1) = 3, \mathsf{sm}_2(2) = 6, \mathsf{sm}_2(3) = 9$ [Kam85] et que $\mathsf{sm}_7(4) = 10$, $\mathsf{sm}_5(4) \leq 11$ [CKO09].

Une autre question importante est de savoir si les n coefficients du *produit court* $\mathrm{FG} \bmod \mathrm{X}^n$ de deux polynômes F et G de degrés au plus n peuvent être calculés plus efficacement que l'ensemble des $2n$ coefficients du produit FG. L'Exercice 2.15, inspiré d'un résultat de Mulders [Mul00], apporte une réponse positive dans le cas où l'algorithme de multiplication est celui de Karatsuba. Par contre, on ne connaît aucun élément de réponse dans le cas où la multiplication polynomiale repose sur la DFT. Une question liée, le calcul du *produit médian* [HQZ04], sera abordée au Chapitre 12.

Une faiblesse de l'algorithme DFT est son comportement quasi-linéaire *par morceaux*, dû aux sauts de complexité entre deux puissances successives de 2. Récemment, van der Hoeven a donné un nouvel algorithme, appelé TFT (de *Truncated Fourier Transform* en anglais), qui coïncide avec la DFT si n est une puissance de 2 et a une courbe de complexité « plus lisse » que la DFT [Hoe04].

Le produit de t polynômes de degrés arbitraires $d_1, \ldots, d_t$ peut s'effectuer [Str83] en $\mathrm{O}(\mathsf{M}(n)(1 + \mathrm{H}(d_1, \ldots, d_t)))$ opérations, où $\mathrm{H}(d_1, \ldots, d_t) = -\sum_i \frac{d_i}{n} \cdot \log \frac{d_i}{n}$ est l'*entropie* du vecteur $(d_1/n, \ldots, d_t/n)$. La multiplication des polynômes à plusieurs variables peut se ramener à une variable grâce à une technique connue sous le nom de *substitution de Kronecker* [Moe76] (voir aussi l'Exercice 5.9).

Bibliographie

Bera Bernstein, Daniel J. (2001). *Multidigit multiplication for mathematicians*. url : http://cr.yp.to/papers.html (visible en 2001).

BL04 Bürgisser, Peter et Martin Lotz (2004). « Lower bounds on the bounded coefficient complexity of bilinear maps ». In : *Journal of the Association for Computing Machinery*, vol. 51, n°3, p. 464–482.

Bri74 Brigham, E. Oran (1974). *The fast Fourier transform*. Prentice-Hall.

BZ07 Bodrato, Marco et Alberto Zanoni (2007). « Integer and polynomial multiplication : towards optimal Toom–Cook matrices ». In : *ISSAC'07 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Dongming Wang. ACM Press, p. 17–24.

CA69 Cook, S. A. et S. O. Aanderaa (1969). « On the minimum computation time of functions ». In : *Transactions of the American Mathematical Society*, vol. 142, p. 291–314.

CH07 Chung, Jaewook et M. Anwar Hasan (2007). « Asymmetric squaring formulae ». In : *ARITH'07 : IEEE Symposium on Computer Arithmetic*. IEEE, p. 113–122.

CK91 Cantor, D. G. et E. Kaltofen (1991). « On fast multiplication of polynomials over arbitrary algebras ». In : *Acta Informatica*, vol. 28, n°7, p. 693–701.

CKO09 Cenk, M., C. K. Koc et F. Ozbudak (2009). « Polynomial multiplication over finite fields using field extensions and interpolation ». In : *ARITH'09 : IEEE Symposium on Computer Arithmetic*. IEEE Computer Society, p. 84–91.

Coo66 Cook, S. (1966). « On the minimum computation time of functions ». Thèse de doctorat. Harvard University.

Coo90 Cooley, James W. (1990). « How the FFT gained acceptance ». In : *A history of scientific computing*. ACM Press Hist. Ser. P : ACM, p. 133–140.

CT65 Cooley, James W. et John W. Tukey (1965). « An algorithm for the machine calculation of complex Fourier series ». In : *Mathematics of Computation*, vol. 19, p. 297–301.

CT93 — (1993). « On the origin and publication of the FFT paper ». In : *Current Contents*, vol. 33, n°51–52, p. 8–9.

De+08 De, A., P. P. Kurur, C. Saha et R. Saptharishi (2008). « Fast integer multiplication using modular arithmetic ». In : *STOC'08 : ACM Symposium on Theory of Computing*. Victoria, British Columbia, Canada : ACM, p. 499–506.

Dem97 Demazure, Michel (1997). *Cours d'algèbre*. Nouvelle Bibliothèque Mathématique n°1. Cassini, Paris.

DS00 Dongarra, J. et F. Sullivan (2000). « Top ten algorithms ». In : *Computing in Science & Engineering*, vol. 2, n°1, p. 22–23.

DV90 Duhamel, P. et M. Vetterli (1990). « Fast Fourier transforms : a tutorial review and a state of the art ». In : *Signal Processing. An Interdisciplinary Journal*, vol. 19, n°4, p. 259–299.

FJ05 Frigo, M. et S. G. Johnson (2005). « The design and implementation of FFTW3 ». In : *Proceedings of the IEEE*, vol. 93, n°2, p. 216–231.

Für07 Fürer, Martin (2007). « Faster integer multiplication ». In : *STOC'07 : ACM Symposium on Theory of Computing*. New York : ACM, p. 57–66.

Für09 — (2009). « Faster integer multiplication ». In : *SIAM Journal on Computing*, vol. 39, n°3, p. 979–1005.

GG03 Gathen, Joachim von zur et Jürgen Gerhard (2003). *Modern computer algebra*. 2e éd. Cambridge University Press.

GS66 Gentleman, W. M. et G. Sande (1966). « Fast Fourier transforms : for fun and profit ». In : *AFIPS'66*. San Francisco, California : ACM, p. 563–578.

HHL14 Harvey, David, Joris van der Hoeven et Grégoire Lecerf (2014). « Faster polynomial multiplication over finite fields ». http://arxiv.org/abs/1407.3361.

HHL16 — (2016). « Even faster integer multiplication ». In : *Journal of Complexity*, vol. 36, p. 1–30.

HJB85 Heideman, Michael T., Don H. Johnson et C. Sidney Burrus (1985). « Gauss and the history of the fast Fourier transform ». In : *Archive for History of Exact Sciences*, vol. 34, n°3, p. 265–277.

Hoe04 Hoeven, Joris van der (2004). « The truncated Fourier transform and applications ». In : *ISSAC'04 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 290–296.

HQZ04 Hanrot, Guillaume, Michel Quercia et Paul Zimmermann (2004). « The middle product algorithm I ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 14, n°6, p. 415–438.

Kam85 Kaminski, Michael (1985). « A lower bound for polynomial multiplication ». In : *Theoretical Computer Science*, vol. 40, n°2-3, p. 319–322.

KO63 Karatsuba, A. et Y. Ofman (1963). « Multiplication of multidigit numbers on automata ». In : *Soviet Physics Doklady*, vol. 7, p. 595–596.

Moe76 Moenck, Robert T. (1976). « Another polynomial homomorphism ». In : *Acta Informatica*, vol. 6, n°2, p. 153–169.

Mon05 Montgomery, Peter L. (2005). « Five, six, and seven-term Karatsuba-like formulae ». In : *IEEE Transactions on Computers*, vol. 54, n°3, p. 362–369.

Mor73 Morgenstern, Jacques (1973). « Note on a lower bound of the linear complexity of the fast Fourier transform ». In : *Journal of the Association for Computing Machinery*, vol. 20, p. 305–306.

Mul00 Mulders, Thom (2000). « On short multiplications and divisions ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 11, n°1, p. 69–88.

Nus80 Nussbaumer, Henri J. (1980). « Fast polynomial transform algorithms for digital convolution ». In : *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, n°2, p. 205–215.

Nus81 — (1981). *Fast Fourier transform and convolution algorithms*. Vol. 2. Springer Series in Information Sciences. Springer-Verlag.

PFM74 Paterson, M. S., M. J. Fischer et A. R. Meyer (1974). « An improved overlap argument for on-line multiplication ». In : *Complexity of computation*. AMS, p. 97–111.

Sch77 Schönhage, A. (1976/77). « Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2 ». In : *Acta Informatica*, vol. 7, n°4, p. 395–398.

SS71 Schönhage, A. et V. Strassen (1971). « Schnelle Multiplikation großer Zahlen ». In : *Computing*, vol. 7, p. 281–292.

Str83 Strassen, V. (1983). « The computational complexity of continued fractions ». In : *SIAM Journal on Computing*, vol. 12, n°1, p. 1–27.

Too63 Toom, A. L. (1963). « The complexity of a scheme of functional elements simulating the multiplication of integers ». In : *Doklady Akademii Nauk SSSR*, vol. 150, p. 496–498.

Van92 Van Loan, Charles (1992). *Computational frameworks for the fast Fourier transform*. Vol. 10. Frontiers in Applied Mathematics. SIAM.

WP06 Weimerskirch, André et Christof Paar (2006). « Generalizations of the Karatsuba algorithm for efficient implementations ». Cryptology ePrint Archive. url : https://eprint.iacr.org/2006/224.pdf.

3. Calculs rapides sur les séries

Résumé

La multiplication rapide de polynômes vue au chapitre précédent permet des calculs efficaces comme l'inversion, l'exponentiation ou la prise de logarithme d'une série. Ces opérations sont effectuées grâce à une version formelle de la méthode de Newton. La composition de séries est en général plus coûteuse que le produit, mais peut aussi tirer parti d'une multiplication rapide.

Ce chapitre porte sur le calcul des premiers termes de développements en séries. Pour fixer les notations, N sera utilisé pour représenter le nombre de termes à calculer (en comptant les coefficients nuls), et « série » sera employé pour « série formelle tronquée à l'ordre N ». Ainsi, « calculer une série » voudra toujours dire en calculer les N premiers termes. Comme dans le Chapitre 2, l'expression $\mathsf{M}(\mathrm{N})$ dénote une borne supérieure sur le nombre d'opérations arithmétiques nécessaires pour multiplier deux polynômes de degré inférieur à N. La fonction M étant une fonction de multiplication polynomiale, rappelons qu'elle est supposée vérifier l'inégalité $\mathsf{M}(\mathrm{N}+\mathrm{N}') \geq \mathsf{M}(\mathrm{N}) + \mathsf{M}(\mathrm{N}')$. L'efficacité des algorithmes sera mesurée par leur complexité arithmétique.

L'opérateur de Newton est très largement utilisé en calcul numérique, pour trouver des solutions approchées d'équations. C'est un processus itératif, qui consiste à chaque étape à remplacer la fonction dont on cherche un zéro par son linéarisé au voisinage de la dernière approximation trouvée. Ainsi, si Φ est une fonction C^1 de $\mathbb{R}$ dans $\mathbb{R}$, pour résoudre $\Phi(y) = 0$ dans $\mathbb{R}$, on choisit $y_0 \in \mathbb{R}$ et on itère

$$y_{k+1} = \mathcal{N}(y_k) = y_k - \frac{\Phi(y_k)}{\Phi'(y_k)}. \tag{3.1}$$

Cette itération est représentée en Figure 3.1. La solution serait exacte si Φ était une fonction linéaire ; dans le cas général, si y_0 est bien choisi (par exemple assez

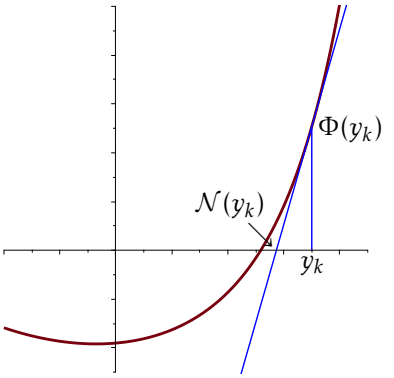


FIGURE 3.1 – Opérateur de Newton de $\mathbb{R}$ dans $\mathbb{R}$.

proche d'une racine simple), l'itération est bien définie ($\Phi'(y_k)$ n'est jamais nul) et la suite y_k converge vers une limite y_∞. En outre, à l'itération k la distance à la limite est de l'ordre du carré de cette distance à l'itération $k-1$. On parle dans ce cas de convergence *quadratique*. Dans ces conditions, pour k suffisamment grand, le nombre de décimales correctes est approximativement *doublé* à chaque itération. De nombreuses généralisations de cette méthode existent, d'abord à des systèmes (la division par l'inverse est alors remplacée par une multiplication à gauche par l'inverse de la matrice jacobienne), puis à des espaces de Banach (où la matrice jacobienne est remplacée par la différentielle de l'opérateur).

Dans le cadre formel, la fonction Φ est remplacée par une application de l'algèbre des séries formelles (définie en Section 3.1) dans elle-même. Cet algèbre n'est pas un espace normé (donc pas un espace de Banach), mais dans bien des cas, l'itération de Newton converge (pour la métrique de la valuation, voir plus loin), et le nombre de *coefficients* corrects est doublé à chaque itération, ce qui mène à une bonne complexité. La composition des séries reste une opération plus coûteuse, à laquelle est consacrée la fin de ce chapitre.

3.1 Séries formelles

Dans cette section et la suivante les séries ont leurs coefficients dans un anneau qui n'est pas nécessairement commutatif, sauf lorsque c'est explicitement indiqué. À partir de la Section 3.3, l'utilisation de la formule de Taylor contraint à se restreindre au cas commutatif.

Définition

Si $\mathbb{A}$ est un anneau (non nécessairement commutatif), on note $\mathbb{A}[[X]]$ l'ensemble des *séries formelles* sur $\mathbb{A}$. Ses éléments sont des suites $(f_i)_{i\in\mathbb{N}}$ de $\mathbb{A}$, notées

$$F(X) = \sum_{i\geq 0} f_i X^i.$$

Le coefficient f_i est appelé le i-ième coefficient de $F(X)$, le coefficient f_0 est appelé terme constant de $F(X)$, et parfois noté $F(0)$. L'indice du premier coefficient non nul est appelé la *valuation* de F et noté $\operatorname{val} F$. Par convention, $\operatorname{val} 0 = \infty$.

Les opérations de $\mathbb{A}[[X]]$ sont l'addition des suites et une multiplication (appelée parfois produit de Cauchy) qui généralise la multiplication des polynômes :

$$\sum_{i\geq 0} f_i X^i \times \sum_{i\geq 0} g_i X^i = \sum_{i\geq 0} h_i X^i, \qquad \text{avec} \quad h_i = \sum_{j+k=i} f_j g_k,$$

la dernière somme étant finie.

Exemple 3.1 La série formelle

$$1 = 1 \cdot X^0 + 0 \cdot X + 0 \cdot X^2 + \cdots,$$

où 1 est l'unité de $\mathbb{A}$, est élément neutre pour la multiplication de $\mathbb{A}[[X]]$. La formule donnant les coefficients du produit se réduit alors à un terme.

Exemple 3.2 Si $F = 1+X$ et $G = 1-X+X^2-X^3+\cdots$, alors le produit vaut $H = FG = 1$: le coefficient de X^n dans H vaut $1-1 = 0$ pour $n > 0$ et 1 sinon. La série G est donc l'inverse de F pour la multiplication, que l'on peut noter $(1+X)^{-1}$.

Métrique

Lorsque les coefficients sont des nombres complexes, la question de la convergence des séries entières représentées par ces séries formelles ne se posera pas pour les algorithmes considérés dans ce chapitre. En revanche, quel que soit l'anneau de coefficients $\mathbb{A}$, il est possible de définir une distance entre deux séries F et G par $d(F,G) = 2^{-\operatorname{val}(F-G)}$.

Exercice 3.1 Vérifier que la fonction d définie ci-dessus est bien une distance. ■

Muni de cette distance, l'ensemble des séries formelles forme un espace métrique *complet* (les suites de Cauchy convergent). En effet, dire qu'une suite (S_n) de séries est de Cauchy signifie qu'étant donné $k \in \mathbb{N}$, il existe un entier K tel que pour tous $m, n \geq K$, on ait $d(S_m, S_n) < 2^{-k}$, autrement dit après l'indice K, les k premiers termes des S_n sont fixés, ce sont ceux de la série limite.

Composition

Si F et G sont deux séries formelles, avec terme constant $G(0) = 0$, on définit la composition comme

$$F(G(X)) = f_0 + f_1 G(X) + f_2 G(X)^2 + \cdots$$

Les points de suspension signifient que l'on considère la limite des sommes H_n obtenues en arrêtant la somme au terme $f_n G(X)^n$. Cette limite existe puisque pour $m \geq n$, la distance obéit à $d(H_m, H_n) \leq 2^{-n}$, ce qui fait de $(H_n)_n$ une suite de Cauchy.

Inverse

Si F est de la forme $F = 1 + XG$ avec $G \in \mathbb{A}[[X]]$, alors F est inversible et son inverse est donné par

$$1 - XG + X^2G^2 - \cdots,$$

composition de $(1+X)^{-1}$ par GX. La preuve découle de cette composition :

$$(1 + XG)(1 - XG + \cdots) = H(XG)$$

où $H = (1+X)(1+X)^{-1} = 1$.

Si $a = F(0)$ est inversible, la série F se récrit $F = a(1 + XG)$ avec $G = a^{-1}(F - a)/X$, donc F est inversible, d'inverse

$$(1 + XG)^{-1}a^{-1} = a^{-1} - Ga^{-1}X + G^2a^{-1}X^2 + \cdots.$$

Ces ingrédients munissent l'ensemble des séries formelles d'une structure d'anneau.

Proposition 3.1 L'ensemble $\mathbb{A}[[X]]$ des séries formelles à coefficients dans $\mathbb{A}$ est un anneau, qui est commutatif si $\mathbb{A}$ l'est. Ses éléments inversibles sont les séries de terme constant inversible.

Démonstration. L'addition est commutative comme celle de $\mathbb{A}$. L'associativité et la distributivité du produit sont obtenues comme pour les polynômes. L'unité pour le produit est la série 1. La première partie de la proposition est donc prouvée.

Si $F = \sum f_i X^i \in \mathbb{A}[[X]]$ est inversible, et $G = \sum g_i X^i$ est son inverse, alors l'extraction du coefficient de X^0 dans l'identité $FG = 1$ donne $f_0 g_0 = 1$ ce qui prouve qu'une série inversible a un terme constant inversible. La réciproque a été présentée ci-dessus. ■

Séries à plusieurs variables

Comme $\mathbb{A}[[X]]$ est un anneau, il peut être utilisé comme anneau de base pour définir des séries formelles en une autre variable Y, ce qui définit de la même manière l'anneau des séries formelles à deux variables $\mathbb{A}[[X]][[Y]]$, noté aussi $\mathbb{A}[[X, Y]]$.

Dérivation

La dérivée d'une série est définie formellement coefficient par coefficient via l'identité

$$\left(\sum_{i\geq 0} f_i X^i\right)' = \sum_{i\geq 0} (i+1) f_{i+1} X^i.$$

Les relations habituelles $(F + G)' = F' + G'$ et $(FG)' = F'G + FG'$ sont prouvées comme pour les polynômes. Dans le cas de séries en plusieurs variables, on utilise la notation des dérivées partielles $\partial F/\partial X$ pour désigner la dérivée par rapport à la variable X.

Troncatures

Algorithmiquement, on ne manipule pas de série à précision « infinie », mais seulement des troncatures, c'est-à-dire un certain nombre des premiers termes. Les séries tronquées deviennent alors de simples polynômes, pour lesquels on dispose en particulier d'algorithmes rapides de multiplication. Étant donnés les N premiers

termes d'une série F, l'objectif de ce chapitre est de donner des algorithmes permettant de calculer efficacement les N premiers termes d'autres séries définies à partir de F.

Pour étendre la notation utilisée avec les polynômes, étant donnée la série

$$S = \sum_{i \geq 0} a_i X^i,$$

on notera S mod X^N le polynôme

$$S \bmod X^N := \sum_{0 \leq i < N} a_i X^i.$$

On notera $S = f + O(X^N)$ si les séries ou polynômes S et f coïncident jusqu'au terme de degré $N-1$, et même plus généralement $S = O(T^k)$ si $S = O(X^{k\,\mathrm{val}(T)})$.

Formule de Taylor

Dans le cas où l'anneau des coefficients $\mathbb{A}$ est commutatif, le lien entre la composition et la dérivation est donné par la formule de Taylor. Si F, G, H sont trois séries formelles, avec $G(0) = H(0) = 0$, et les entiers $2, 3, \ldots, k-1$ sont inversibles dans $\mathbb{A}$, alors

$$F(G+H) = F(G) + F'(G)H + F''(G)\frac{H^2}{2!} + \cdots + O(H^k).$$

La formule est classique pour les polynômes, et les coefficients de X^N dans les deux membres de la formule sont les mêmes que ceux de l'identité entre polynômes obtenue en considérant les troncatures de F, G et H modulo X^{N+1}, ce qui permet de conclure.

Pour cette identité, la commutativité de $\mathbb{A}$ est cruciale comme le montre l'exemple de $F = X^2$: on a alors $(G+H)^2 = G^2 + GH + HG + H^2$ et la partie linéaire $GH + HG$ est en général différente de $2GH = F'(G)H$.

Intégration

Il s'agit de l'opération inverse de la dérivation. On suppose que les entiers sont inversibles dans $\mathbb{A}$ ($\mathbb{A}$ est une $\mathbb{Q}$-algèbre) et on définit alors

$$\int \sum_{i \geq 0} f_i X^i = \sum_{i \geq 0} f_i \frac{X^{i+1}}{i+1}.$$

Exemple 3.3 La série $\log(1+X)$ vaut

$$\log(1+X) = \int (1+X)^{-1} = X - \frac{1}{2}X^2 + \frac{1}{3}X^3 + \cdots$$

3.2 La méthode de Newton pour le calcul d'inverses

Partant de la fonction $\Phi(y) = 1/y - f$, dont l'unique racine est l'inverse $1/f$, l'itération (3.1) devient

$$\mathcal{N}(g_k) = g_k + g_k^2(1/g_k - f) = g_k + g_k(1 - g_k f).$$

Cette itération permet le calcul d'inverses dans plusieurs contextes.

Entrée Un entier $N > 0$, $F \bmod X^N$ une série tronquée.
Sortie $F^{-1} \bmod X^N$.
Si $N = 1$, alors renvoyer f_0^{-1}, où $f_0 = F(0)$.
Sinon :
1. Calculer récursivement l'inverse G de F mod $X^{\lceil N/2 \rceil}$.
2. Renvoyer $G + (1 - GF)G \bmod X^N$.

Algorithme 3.2 – Inverse de série par itération de Newton.

Convergence quadratique pour l'inverse d'une série formelle

Lemme 3.2 Soient $\mathbb{A}$ un anneau non nécessairement commutatif, $F \in \mathbb{A}[[X]]$ une série formelle de terme constant inversible et G une série telle que $G - F^{-1} = O(X^n)$ ($n \geq 1$). Alors la série

$$\mathcal{N}(G) = G + (1 - GF)G \tag{3.2}$$

vérifie $\mathcal{N}(G) - F^{-1} = O(X^{2n})$.

Démonstration. Par hypothèse, on peut définir $H \in \mathbb{A}[[X]]$ par $1 - GF = X^n H$. Il suffit alors de récrire $F = G^{-1}(1 - X^n H)$ et d'inverser :

$$F^{-1} = (1 + X^n H + O(X^{2n}))G = G + X^n HG + O(X^{2n})G = \mathcal{N}(G) + O(X^{2n}). \qquad \blacksquare$$

Algorithme

Lemme 3.3 L'Algorithme 3.2 d'inversion est correct.

Démonstration. La preuve est une récurrence sur les entiers. Pour $N = 1$ la propriété est claire. Si la propriété est vraie pour $1 \leq N < k$, alors elle l'est pour $1 \leq \lceil N/2 \rceil < k$ donc le résultat de l'étape (1) vaut $F \bmod X^{\lceil N/2 \rceil}$. Alors, d'après le Lemme 3.2, $G + (1 - GF)G = F^{-1} \bmod X^{2\lceil N/2 \rceil}$ et la troncature modulo X^N est donc bien correcte. $\blacksquare$

Complexité

Chaque itération coûte essentiellement deux multiplications, et l'ensemble est donc effectué en $O(M(N))$ opérations d'après le lemme « diviser pour régner ». Une estimation plus précise est donnée par la proposition suivante.

Proposition 3.4 Soient $\mathbb{A}$ un anneau non nécessairement commutatif, F dans $\mathbb{A}[[X]]$ avec $F(0)$ inversible et $N \geq 1$. On peut calculer l'inverse de F modulo X^N en $3M(N) + O(N)$ opérations dans $\mathbb{A}$ lorsque N est une puissance de 2.

Démonstration. L'étape (2) de l'algorithme demande deux multiplications : comme G est de degré inférieur à $\lceil N/2 \rceil$, celle de G par F coûte $2M(\lceil N/2 \rceil)$ et produit un résultat de la forme $X^{\lceil N/2 \rceil} R_N$ avec R_N de degré inférieur à $\lceil N/2 \rceil$. Sa multiplication par G

coûte alors $M(\lceil N/2 \rceil)$ opérations. S'ajoutent à ces multiplications $\lambda \cdot N$ opérations supplémentaires (additions, ...), λ étant une constante que l'on ne va pas déterminer. Notons $C(N)$ le coût du calcul modulo N ; on a alors

$$C(N) \leq C(\lceil N/2 \rceil) + 3M(\lceil N/2 \rceil) + \lambda N.$$

L'application du lemme « diviser pour régner » donne alors

$$C(N) \leq 3M(N) + 2\lambda N$$

pour N puissance de 2 en utilisant l'inégalité $M(d) \leq \frac{1}{2}M(2d)$. ■

Comme d'habitude, lorsque N n'est pas une puissance de 2, la complexité est bornée par celle de la puissance de 2 immédiatement supérieure, ce qui rajoute au plus un facteur 2.

Optimisations

Le coût de la Proposition 3.4 peut être encore amélioré. Le produit GF est de la forme $1 + X^{\lceil N/2 \rceil} R_N$. Il s'agit donc de calculer le produit d'un polynôme de degré N/2 par un polynôme de degré N, alors que l'on connaît à l'avance les N/2 premiers coefficients du produit. Dans cette situation, il est possible d'abaisser le coût du produit de $M(N)$ à $M(N/2)$, en utilisant un produit médian et des techniques non triviales de « transposition » de code abordées au Chapitre 12.

Division de séries

Le quotient H/F où H et F sont des séries et F est inversible, peut être obtenu comme le produit $H \times F^{-1}$. Ce calcul peut être amélioré d'un facteur constant. Il suffit pour cela de remplacer la dernière itération $G + (1 - GF)G$ par $Y + (H - YF)G$ où $Y := HG \bmod X^{\lceil N/2 \rceil}$. Cette astuce remplace un produit de la forme $(N/2, N)$ (pour GF), un produit de la forme $(N/2, N/2)$ (pour $(1 - GF)G$) et un produit final (N, N) par deux produits $(N/2, N/2)$ (pour Y et $(H - YF)G$) et un produit $(N/2, N)$ pour calculer YF. L'économie est donc la différence $M(N) - M(N/2)$.

Application à la division euclidienne

L'Algorithme 3.2 est l'élément clé de la division euclidienne rapide, traitée au Chapitre 4.

Application au logarithme

Si $F \in \mathbb{A}[[X]]$ est telle que $F(0) = 0$, on définit la série $\log(1 + F)$ par composition avec $\log(1 + X)$. Pour calculer cette série lorsque $\mathbb{A}$ est commutatif, il suffit d'utiliser l'identité

$$\log(1 + F) = \int \frac{F'}{1 + F}.$$

Exercice 3.2 Prouver cette identité à partir des définitions de log, $\int$ et de la dérivation données plus haut. ■

Le calcul demande une division de séries, une dérivation et une intégration, mais ces deux dernières opérations sont de complexité linéaire. Le bilan est donc une complexité en $O(M(N))$.

Inverse de matrices

L'anneau $\mathbb{A}$ n'étant pas supposé commutatif, il est possible de traiter directement des matrices de séries comme des séries de matrices : le Lemme 3.2 et l'Algorithme 3.2 s'appliquent. La complexité de cet algorithme s'exprime à l'aide de la fonction $\mathsf{MM}(k,N)$ complexité du produit de matrices $k\times k$ de polynômes de degré au plus N. Le Corollaire 5.11 du Chapitre 5 montre que $\mathsf{MM}(k,N) = O(k^\theta N + k^2 M(N))$.

Proposition 3.5 Soient $\mathbb{A}$ un anneau non nécessairement commutatif, A une matrice $k \times k$ à coefficients dans $\mathbb{A}[[X]]$, telle que $A(0)$ soit inversible. La matrice $A^{-1} \bmod X^N$ peut être calculée en $O(\mathsf{MM}(k,N))$ opérations dans $\mathbb{A}$.

Démonstration. Comme pour l'inverse dans l'Algorithme 3.2, le coût est dominé à une constante près par celui du calcul de $G+(I-GF)G \bmod X^N$: la complexité vérifie l'inégalité

$$C(N) \le C(\lceil N/2 \rceil) + 2\mathsf{MM}(k,N)$$

et le résultat se déduit de l'hypothèse $\mathsf{MM}(k,N) \le \frac{1}{2}\mathsf{MM}(k,2N)$. ■

3.3 Itération de Newton formelle et applications

Dans toute la suite de ce chapitre, l'anneau $\mathbb{A}$ est supposé commutatif.

Un résultat général

Théorème 3.6 — Itération de Newton sur les séries. Soit $\Phi \in \mathbb{A}[[X,Y]]$ une série à deux variables en X et Y, telle que $\Phi(0,0) = 0$ et $\frac{\partial\Phi}{\partial Y}(0,0)$ est inversible dans $\mathbb{A}$. Il existe alors une unique série $S \in \mathbb{A}[[X]]$, telle que $\Phi(X,S) = 0$ et $S(0) = 0$. Si F est une série telle que $S - F = O(X^n)$ $(n \ge 1)$, alors

$$\mathcal{N}(F) = F - \frac{\Phi(X,F)}{\frac{\partial\Phi}{\partial Y}(X,F)}$$

vérifie $S - \mathcal{N}(F) = O(X^{2n})$.

Ce résultat est un résultat local en $(0,0)$. Il exprime que si le point $(0,0)$ est solution, il s'étend en une courbe solution. Par translation, d'autres situations où la solution n'est pas en $(0,0)$ s'y ramènent. La première partie est une version du théorème des fonctions implicites pour les séries formelles.

Démonstration. D'abord il faut observer que l'itération est bien définie : la composition de Φ et de $\frac{\partial\Phi}{\partial Y}$ avec F sont possibles parce que $F(0) = S(0) = 0$. Pour la même raison,

Entrée Un entier $N > 0$, Φ mod (X^N, Y^N) une série tronquée avec $\Phi(0,0) = 0$ et $(\partial\Phi/\partial Y)(0,0)$ inversible.
Sortie S mod X^N, T mod $X^{\lceil N/2\rceil}$, telles que $S(0) = 0$, $\Phi(X,S) = 0$, et $T = (\partial\Phi/\partial Y(X,S))^{-1}$.

1. Si $N = 1$, alors renvoyer 0, $((\partial\Phi/\partial Y)(0,0))^{-1}$.
2. Sinon :
   a. Calculer récursivement $F := S \bmod X^{\lceil N/2\rceil}$ et $G := T \bmod X^{\lceil\lceil N/2\rceil/2\rceil}$.
   b. Calculer $G := G + (1 - G\frac{\partial\Phi}{\partial Y}(X,F))G \bmod X^{\lceil N/2\rceil}$.
   c. Calculer $F := F - G\Phi(X,F) \bmod X^N$.
   d. Renvoyer F, G.

Algorithme 3.3 – Résolution de $\Phi(X,Y) = 0$ par itération de Newton.

$\frac{\partial\Phi}{\partial Y}(X,F)$ est inversible puisque son terme constant $\frac{\partial\Phi}{\partial Y}(0,0)$ est inversible. De ce fait, on déduit aussi $\mathrm{val}(\mathcal{N}(F) - F) = \mathrm{val}(\Phi(X,F))$.

Ensuite, la preuve se déroule en deux temps : l'itération est d'abord utilisée pour montrer l'existence d'une série solution S, puis on montre l'unicité en même temps que la vitesse de convergence vers S.

La valuation de $\Phi(X,F)$ est doublée par l'itération : la formule de Taylor donne

$$\Phi(X,\mathcal{N}(F)) = \Phi(X,F) + \frac{\partial\Phi}{\partial Y}(X,F)(\mathcal{N}(F) - F) + O((\mathcal{N}(F) - F)^2) = O(\Phi(X,F)^2). \tag{3.3}$$

La suite définie par $S_0 = 0$ et $S_{k+1} = \mathcal{N}(S)$ vérifie donc

$$\mathrm{val}(S_{k+2} - S_{k+1}) = \mathrm{val}(\Phi(X,S_{k+1})) \geq 2\,\mathrm{val}(S_{k+1} - S_k).$$

Cette suite est donc de Cauchy et sa limite S existe et vérifie $\Phi(X,S) = \lim(\Phi(X,S_k)) = 0$. La vitesse de convergence vers S vient encore de la formule de Taylor. En effet, l'égalité

$$\Phi(X,F) = \Phi(X,S) + \frac{\partial\Phi}{\partial Y}(X,F)(S - F) + O((S - F)^2) \tag{3.4}$$

donne $S - F = O(\Phi(X,F))$. On en déduit $S - \mathcal{N}(F) = O(\Phi(X,\mathcal{N}(F))) = O(X^{2n})$ au vu des hypothèses et de (3.3). Cette égalité donne aussi l'unicité de la solution : si F est une autre solution avec $F(0) = 0$, elle doit vérifier $S - F = O((S - F)^2)$, ce qui entraîne $\mathrm{val}(S - F) = \infty$ et donc $F = S$. ■

Algorithme

Lemme 3.7 L'Algorithme 3.3 de résolution est correct.

Démonstration. Pour $N = 1$, les valeurs renvoyées sont les termes constants de S et T. Si la propriété est vraie pour $1 \leq k < N$, alors l'étape (1.a) renvoie T mod $X^{\lceil\lceil N/2\rceil/2\rceil}$ et

S mod $X^{\lceil N/2\rceil}$. D'après le Lemme 3.2, l'étape (1.b) calcule T mod $X^{\lceil N/2\rceil}$. L'étape (1.c) calcule alors S mod X^N : les formules (3.3) et (3.4) montrent que seuls les $\lceil N/2\rceil$ premiers coefficients de $\partial\Phi/\partial Y(X,S)$ sont utiles pour doubler la précision. ■

Applications

Un exemple d'application est fourni par l'inverse de la section précédente avec $\Phi(X,Y) = f - 1/(f(0)^{-1} + Y)$ (lorsque $\mathbb{A}$ est commutatif). La complexité de l'Algorithme 3.3 dépend en particulier de celle du calcul de Φ et de $\partial\Phi/\partial Y$. Les applications les plus directes de la méthode de Newton sont résumées dans les théorèmes suivants.

Théorème 3.8 Dans le cas où la série Φ du Théorème 3.6 est un polynôme en Y de degré O(1), les N premiers termes de sa série solution S, telle que $S(0) = 0$, sont obtenus en O(M(N)) opérations dans $\mathbb{A}$.

Démonstration. L'argument est le même que pour la complexité de l'inverse. Pour Φ polynôme de degré O(1), les évaluations de Φ et de Φ' coûtent O(M(N)), le reste de l'argument est inchangé. ■

Dans le cas où Φ est un polynôme à la fois en X et en Y, un algorithme de complexité O(N) (donc linéaire au lieu de quasi-linéaire, et ce, quelle que soit la multiplication utilisée) sera présenté au Chapitre 14.

Théorème 3.9 Soit f une série de $\mathbb{A}[[X]]$. On peut calculer en O(M(N)) opérations dans $\mathbb{A}$ les N premiers termes de :
1. f^{-1}, lorsque $f(0)$ est inversible ;
2. $\log f$ et f^α lorsque $f(0) = 1$ et $\alpha \in \mathbb{K}$;
3. $\exp(f)$ lorsque $f(0) = 0$.

Pour les points (2) et (3), nous supposons également que $2, 3, \ldots, N-1$ sont des éléments inversibles de $\mathbb{A}$.

Démonstration. La preuve pour l'inverse et le logarithme provient des Sections 3.2 et 3.2. L'exponentielle est traitée dans la section qui vient, et la puissance se déduit des précédentes par

$$f^\alpha = \exp(\alpha \log f).$$

Une application remarquable de cette identité évidente apparaît dans la Section 3.3 sur l'inverse compositionnel. ■

Exponentielle

Si $F \in \mathbb{A}[[X]]$ est telle que $F(0) = 0$, on définit la série exp(F) par

$$\exp(F) = 1 + F + F^2/2! + F^3/3! + \cdots$$

lorsque $2, 3, \ldots$ sont inversibles dans $\mathbb{A}$. Pour calculer cette série, l'idée est d'appliquer une méthode de Newton avec $\Phi(Y) = F - \log Y, \quad Y(0) = 1$.

Entrée Un polynôme $P \in \mathbb{K}[X]$ de degré d, un entier k.
Sortie Les sommes de Newton $p_0, \ldots, p_k$ de P.
1. Former les polynômes $N = T^{d-1}P'(1/T)$ et $D = T^d P(1/T)$.
2. Calculer $N/D \bmod T^{k+1} = p_0 + p_1 T + \cdots + p_k T^k$.
3. Renvoyer les coefficients.

Algorithme 3.4 – Calcul des sommes de Newton à partir des coefficients.

Entrée Les sommes de Newton $p_0, \ldots, p_d$ du polynôme unitaire $P \in \mathbb{K}[X]$.
Sortie Les coefficients de P.
1. Former la série $S_1 := p_1 + p_2 T + \cdots + p_d T^{d-1}$.
2. Calculer $S_2 := \exp(-\int S_1) \bmod T^{d+1}$.
3. Renvoyer $X^{p_0} S_2(1/X)$.

Algorithme 3.5 – Calcul des coefficients à partir des sommes de Newton.

Exercice 3.3 Montrer que les conditions du Théorème 3.6 sont vérifiées. ∎

Il s'ensuit une convergence quadratique vers $\exp(F)$ pour l'itération

$$E_{k+1} = \mathcal{N}(E_k) = E_k + E_k(F - \log E_k).$$

Chaque itération demande donc le calcul d'une multiplication et d'un logarithme, d'où la complexité en $O(M(N))$. Une autre itération légèrement plus efficace est présentée en Exercice 3.8.

Exercice 3.4 Donner une itération de Newton qui calcule directement la racine carrée, sans passer par l'exponentielle et le logarithme. Plus difficile, estimer le gain par rapport à l'algorithme général de calcul de puissance. ∎

Exercice 3.5 Donner des bornes sur les constantes dans les $O(\cdot)$ pour le logarithme, l'exponentielle et la puissance. ∎

Sommes de Newton

Une conséquence intéressante de l'efficacité du calcul de l'exponentielle est l'utilisation efficace des *sommes de Newton* comme structure de données.

Si $P \in \mathbb{K}[X]$ est un polynôme de degré d, avec $\mathbb{K}$ un corps, alors P a d racines $\alpha_1, \ldots, \alpha_d$ dans la clôture algébrique $\bar{\mathbb{K}}$ de $\mathbb{K}$. Rappelons qu'un polynôme est dit *unitaire* lorsque son coefficient de plus haut degré est 1. Les *sommes de Newton* de P sont les sommes $p_i = \alpha_1^i + \cdots + \alpha_d^i$. Ce sont des éléments de $\mathbb{K}$, puisqu'ils sont fonctions symétriques des racines de P. (Tout polynôme symétrique en les racines d'une équation algébrique s'exprime comme un polynôme en les coefficients de l'équation.) Leur utilisation efficace repose sur la proposition suivante.

Proposition 3.10 Les sommes de Newton $p_1, \ldots, p_d$ d'un polynôme $P \in \mathbb{K}[X]$ de degré d peuvent être calculées par l'Algorithme 3.4 en $O(M(d))$ opérations dans $\mathbb{K}$. Lorsque la caractéristique de $\mathbb{K}$ est 0 ou strictement supérieure à d, la conversion inverse est également possible en $O(M(d))$ opérations par l'Algorithme 3.5.

Démonstration. Si $P = c\prod_\alpha (X - \alpha)$, la décomposition en éléments simples

$$S = \frac{XP'}{P} = \sum_{P(\alpha)=0} \frac{X}{X-\alpha} = \sum_{\substack{P(\alpha)=0 \\ i\geq 0}} \alpha^i X^{-i},$$

où α décrit les zéros de P comptés avec multiplicité, montre que les coefficients du développement en puissances de X^{-1} de XP'/P sont les p_i. Pour se ramener à des calculs de séries formelles, la première étape introduit une nouvelle variable par $X = 1/T$. Le calcul de la dérivée a une complexité linéaire en d ; les conversions en puissances de T ne demandent pas d'opérations arithmétiques. Les $k = d$ premiers termes de la division sont obtenus $O(M(d))$ opérations, comme montré en Section 3.2.

L'opération inverse, à savoir le calcul de P à partir des d premiers termes de la série $S = XP'/P$ est effectué en $O(M(d))$ opérations par la formule $P = \exp \int S/X$. À nouveau, il est plus simple de travailler en T pour rester dans le domaine des séries formelles. Le signe '-' dans le calcul de S_2 provient du signe de l'intégration des puissances négatives de X. La contrainte sur la caractéristique permet de définir l'exponentielle tronquée par son développement en série. ■

Application à la somme et au produit composés

On suppose dans cette section que $\mathbb{K}$ est un corps de caractéristique nulle ou strictement supérieure à d.

Théorème 3.11 Soient P et Q deux polynômes unitaires de $\mathbb{K}[X]$, de degrés d_p et d_q et soit $d = d_p d_q$. Alors, les polynômes

$$P \oplus Q = \prod_{P(\alpha)=0,\ Q(\beta)=0} \big(X - (\alpha + \beta)\big), \qquad P \otimes Q = \prod_{P(\alpha)=0,\ Q(\beta)=0} (X - \alpha\beta)$$

peuvent être calculés en $O\big(M(d)\big)$ opérations dans $\mathbb{K}$ par les Algorithmes 3.6 et 3.7.

Comme ces polynômes sont de degré d, il s'agit là encore d'une complexité quasi-optimale. Dans cet énoncé, les polynômes sont définis à l'aide des racines de P et Q dans une clôture algébrique, mais leurs coefficients sont dans $\mathbb{K}$. Il est également possible de les définir sur un anneau quelconque comme des *résultants* en deux variables. Les résultants et leur calcul sont présentés au Chapitre 6. En général, avec deux variables, il n'existe pas encore d'algorithme quasi-optimal pour le résultant et le résultat ci-dessus est l'un des rares cas particuliers qui se traite efficacement.

Les polynômes constituent une structure de données bien commode pour coder des nombres algébriques et calculer avec : les polynômes $P \oplus Q$ et $P \otimes Q$ fournissent l'addition et la multiplication avec cette structure de données.

Entrée Des polynômes unitaires P et Q de $\mathbb{K}[X]$, de degrés d_p et d_q.
Sortie Leur produit composé $P \otimes Q$ de degré $d := d_p d_q$.
1. Calculer les sommes de Newton $(p_0, \dots, p_{d^2})$ et $(q_0, \dots, q_{d^2})$ de P et Q.
2. Renvoyer le polynôme dont les sommes de Newton sont $(p_0 q_0, \dots, p_{d^2} q_{d^2})$.

Algorithme 3.6 – Produit composé.

Entrée les polynômes unitaires P et Q de $\mathbb{K}[X]$, de degrés d_p et d_q.
Sortie leur somme composée $P \oplus Q$ de degré $d := d_p d_q$.
1. Calculer les sommes de Newton $(p_0, \dots, p_{d^2})$ et $(q_0, \dots, q_{d^2})$ de P et Q.
2. Former les polynômes
$$S_p := p_0 + p_1 \frac{T}{1!} + \cdots + p_{d^2} \frac{T^{d^2}}{(d^2)!},\ S_q := q_0 + q_1 \frac{T}{1!} + \cdots + q_{d^2} \frac{T^{d^2}}{(d^2)!}.$$
3. Calculer $S := S_p \times S_q \bmod T^{d^2+1} = s_0 + s_1 T + s_2 T^2 + \cdots$
4. Renvoyer le polynôme dont les sommes de Newton sont $(s_0, 1!s_1, 2!s_2, \dots, (d^2)!s_{d^2})$.

Algorithme 3.7 – Somme composée.

Démonstration. Le produit de sommes de Newton est la somme de Newton du produit :
$$\sum_{\alpha} \alpha^i \sum_{\beta} \beta^i = \sum_{\alpha,\beta} (\alpha\beta)^i.$$
Il suffit donc de multiplier *terme à terme* les séries génératrices des sommes de Newton de P et Q pour obtenir la série génératrice des sommes de Newton de $P \otimes Q$. Si l'on a calculé d termes de ces séries, le résultant, dont le degré est d, est alors reconstruit par une exponentielle en $O(M(d))$ opérations.

Diviser le i-ième coefficient de XP'/P par $i!$, pour tout $i \geq 0$, produit la série
$$\sum_{\substack{P(\alpha)=0 \\ i \geq 0}} \frac{\alpha^i X^{-i}}{i!} = \sum_{P(\alpha)=0} \exp(\alpha/X).$$
En effectuant la même opération sur Q et en multipliant les séries, on obtient donc
$$\sum_{P(\alpha)=0} \exp(\alpha/X) \sum_{Q(\beta)=0} \exp(\beta/X) = \sum_{\substack{P(\alpha)=0 \\ Q(\beta)=0}} \exp((\alpha+\beta)/X).$$

Il suffit alors de remultiplier le i-ième coefficient par $i!$ pour obtenir la série génératrice des sommes de Newton du polynôme $P \otimes Q$, qui est reconstruit par une exponentielle. L'ensemble des opérations a une complexité bornée par $O(M(d))$. ■

Systèmes

Le Théorème 3.6 s'étend à des systèmes d'équations.

Théorème 3.12 Soient $\Phi = (\Phi_1, \ldots, \Phi_k)$ des séries en $k+1$ indéterminées X et $Y = (Y_1, \ldots, Y_k)$, telles que $\Phi(0) = 0$ et la matrice jacobienne $(\frac{\partial \Phi}{\partial Y})$ est inversible dans $\mathbb{A}$ en 0. Alors, le système

$$\Phi_1(X, Y_1, \ldots, Y_k) = \cdots = \Phi_k(X, Y_1, \ldots, Y_k) = 0$$

admet une solution $S = (S_1, \ldots, S_k) \in \mathbb{A}[[X]]^k$ telle que $S(0) = 0$. Si $F = (F_1, \ldots, F_k)$ est tel que $S - F = O(X^n)$ $(n \geq 1)$, alors

$$\mathcal{N}(F) = \begin{pmatrix} F_1 \\ \vdots \\ F_k \end{pmatrix} - \left(\frac{\partial \Phi}{\partial Y}(X, F_1, \ldots, F_k)\right)^{-1} \cdot \begin{pmatrix} \Phi_1(X, F_1, \ldots, F_k) \\ \vdots \\ \Phi_k(X, F_1, \ldots, F_k) \end{pmatrix}$$

vérifie $S - \mathcal{N}(F) = O(X^{2n})$.

La preuve est la même que celle du Théorème 3.6, la formule de Taylor pour les fonctions de plusieurs variables s'exprimant à l'aide de la matrice jacobienne.

Lemme 3.13 L'Algorithme 3.3 fonctionne aussi, sans changement, pour un système Φ satisfaisant aux hypothèses du Théorème 3.12.

Démonstration. La preuve est la même, en observant que l'étape (1.b) est écrite de telle sorte que le Lemme 3.2 s'applique : cette étape calcule l'inverse de la matrice jacobienne à la précision requise et la dernière étape effectue un produit matrice-vecteur pour en déduire la correction de la nouvelle itération. ■

Le corollaire suivant joue un rôle important dans l'algorithme de résolution géométrique du Chapitre 28.

Corollaire 3.14 — Systèmes polynomiaux. Sous les mêmes hypothèses, si $\Phi_i \in \mathbb{A}[[X]][Y_1, \ldots, Y_k]$ pour $1 \leq i \leq k$, alors les solutions séries à précision N peuvent être calculées en $O(MM(k, N))$ opérations dans $\mathbb{A}$.

Démonstration. Dans ce calcul, le nombre d'opérations nécessaires à l'évaluation des polynômes ou de leurs dérivées en une série à précision N est borné par une constante fois $MM(N)$, constante qui ne dépend que des polynômes Φ_i. Ensuite, l'étape (b.) demande des produits de matrices $k \times k$ de polynômes de degré $\lceil N/2 \rceil$, et l'étape (c.) demande un produit matrice-vecteur en précision N. La complexité de l'application

récursive est donc dominée par celle de l'étape (2), le résultat est donc le même que pour l'inverse de matrices de séries. ■

Inverse compositionnel

Étant donnée une série F avec $F(0) = 0$ et $F'(0)$ inversible, il s'agit ici de calculer une série $F^{(-1)}$ telle que $F(F^{(-1)}(X)) = F^{(-1)}(F(X)) = X$.

Les N premiers termes

Les conditions d'application de la méthode de Newton (Théorème 3.6) sont vérifiées pour la fonction

$$\Phi(X, Y) = F(Y) - X.$$

L'opérateur de Newton correspondant est donc

$$\mathcal{N}(G) = G - \frac{F(G) - X}{F'(G)}.$$

La dérivation de $F(G) = X$ donne $F'(G)G' = 1$ et cet opérateur se simplifie en

$$\mathcal{N}(G) = G - G' \times (F(G) - X).$$

Le coût est alors dominé soit par celui du produit, soit plus généralement par celui de la composition par F (voir Section 3.4).

Exercice 3.6 La k-ième racine positive de l'équation $x = \tan x$ admet un développement asymptotique de la forme

$$r_k = (2k+1)\frac{\pi}{2} + \frac{a_1}{k} + \frac{a_2}{k^2} + \cdots.$$

Pour tout i, le coefficient a_i s'écrit $a_i = f_i(1/\pi)$, où f_i est un polynôme impair de degré $2i+1$. Borner le nombre d'opérations dans $\mathbb{Q}$ nécessaires pour calculer $f_1, \ldots, f_N$, pour N grand. ■

Exemple

La série génératrice exponentielle $T \in \mathbb{Q}[[X]]$ des arbres généraux étiquetés (appelés aussi *arbres de Cayley*) vérifie l'équation

$$T = X\exp(T),$$

c'est-à-dire que le coefficient de X^n dans la série T est le nombre d'arbres de taille n divisé par $n!$. Ces coefficients sont obtenus en $O(M(N))$ opérations dans $\mathbb{Q}$ en calculant d'abord ceux de $Y\exp(-Y)$ par les méthodes de la section précédente, puis en inversant ce développement par la méthode présentée ci-dessus.

Pour déterminer le comportement asymptotique des coefficients ainsi calculés lorsque N tend vers l'infini (autrement dit, l'asymptotique du nombre d'arbres de taille N), une méthode utilisée en analyse d'algorithmes passe par le calcul du développement de T au voisinage du point singulier $\exp(-1) = 1/e$, où T vaut 1. En posant $u = \sqrt{2}\sqrt{1 - eX}$, on voit qu'il s'agit alors de calculer le développement de y solution de

$$\sqrt{2 - 2y\exp(1-y)} = u,$$

au voisinage de $y = 1$ et $u = 0$, équation à laquelle la méthode présentée dans cette section s'applique pour donner le résultat en $O(M(N))$ opérations.

Le N-ième terme

Bien qu'en général le coût de l'inverse compositionnel soit dominé par celui de la composition et donc en $O(\sqrt{N \log N} M(N))$ (voir la section suivante), il est possible d'obtenir le N-ième terme sans calculer les précédents pour $O(M(N))$ opérations. L'idée repose sur la *formule d'inversion de Lagrange* :

$$[X^N]F^{(-1)}(X) = \frac{1}{N}[X^{N-1}]\frac{1}{(F(X)/X)^N}, \tag{3.5}$$

où la notation $[X^k]F$ représente le coefficient de X^k dans la série F. Ainsi, le calcul par cette formule ne requiert qu'une puissance et donc peut être effectué en $O(M(N))$ opérations.

R Cette formule donne immédiatement une jolie expression pour les coefficients de T solution de $T = X\exp(T)$ de l'exemple précédent : le n-ième coefficient vaut $n^{n-1}/n!$. Cette expression permet de calculer les N premiers coefficients en $O(N)$ opérations, ce qui est légèrement mieux que le $O(M(N))$ du cas général. L'asymptotique mentionnée plus haut redonne dans cet exemple la formule de Stirling.

3.4 La composition des séries

Si $F(X)$ et $G(X)$ sont des séries, avec $G(0) = 0$, il s'agit de calculer efficacement la série $F(G(X))$. À la différence des algorithmes vus dans les sections précédentes, il s'avère que les algorithmes connus n'atteignent pas une complexité quasi-optimale (c'est-à-dire linéaire en N, à des facteurs logarithmiques près).

Méthode naïve

La composition peut être calculée par la méthode de Horner. Si $F(X) = \sum_{i \geq 0} f_i X^i + O(X^N)$, le calcul utilise alors

$$F(G(X)) = f_0 + G \times (f_1 + G \times (f_2 + \cdots)) + O(G^N)$$

et $O(G^N) = O(X^N)$.

Exercice 3.7 Montrer que cette formule amène à un coût $O(NM(N))$. ■

Pas de bébés, pas de géants

Une technique dite « pas de bébés, pas de géants » réduit ce coût à $O(\sqrt{N}(M(N) + MM(\sqrt{N}))) = O(\sqrt{N}(M(N) + N^{\theta/2}))$, où θ est un exposant réalisable pour la complexité du produit de matrices, présenté au Chapitre 8. La série F est d'abord écrite

$$F(X) = F_0(X) + X^k F_1(X) + \cdots + X^{k\lceil N/k \rceil} F_{\lceil N/k \rceil}(X) + O(X^N),$$

où les polynômes F_i, en nombre $1+\lceil N/k\rceil$, ont un degré au plus $k-1$. Ensuite, on calcule les puissances $G^2,\dots,G^k$ de G en $k\mathsf{M}(N)$ opérations. Soient $f_{i,j}$ les coefficients des F_i et $g_{i,j}$ ceux des G^i. Le développement

$$F_i(G)=\sum_{j=0}^{N-1}\left(\sum_{\ell=0}^{k-1} f_{i,\ell}g_{\ell,j}\right)X^j+O(X^N)$$

montre que les coefficients des $F_i(G)$ sont les coefficients du produit de la matrice $(f_{i,j})$ de taille $(\lceil N/k\rceil+1)\times k$ par la matrice $(g_{i,j})$ de taille $k\times N$. En découpant chacune de ces matrices en blocs de taille $k\times k$ — la première devient un vecteur colonne d'environ N/k^2 blocs, la seconde un vecteur ligne d'environ N/k blocs —, ce produit est effectué en au plus $O(N^2k^{\theta-3})$ opérations. Ensuite, il ne reste plus qu'à effectuer $\lceil N/k\rceil$ produits à précision N suivis d'autant d'additions pour un coût total de $O(N\mathsf{M}(N)/k)$. Le coût total est minimisé par le choix $k=\lfloor\sqrt{N}\rfloor$ qui mène à la complexité annoncée.

L'algorithme de Brent–Kung

Bien qu'il n'y ait pas d'algorithme permettant d'abaisser la complexité de la composition au même ordre que $\mathsf{M}(N)$, il est possible de faire sensiblement mieux que la méthode précédente, en évitant la multiplication matricielle, grâce à l'Algorithme 3.8 sophistiqué, dû à Brent et Kung.

Théorème 3.15 Si $2,3,\dots,\lceil\sqrt{N\log N}\rceil$ sont inversibles dans $\mathbb{A}$ et si F(X) et G(X) sont deux séries de $\mathbb{A}[[X]]$ avec $G(0)=0$ et $G'(0)$ inversible, alors l'Algorithme 3.8 calcule la série $F(G(X))$ mod X^N en $O(\sqrt{N\log N}\mathsf{M}(N))$ opérations arithmétiques, ou en $O(\sqrt{N}\mathsf{M}(N))$ opérations arithmétiques si $\log\mathsf{M}(N)/\log N\to\gamma>1$.

L'idée de départ de l'algorithme permettant d'aboutir à cette complexité est d'utiliser la formule de développement de Taylor, sous la forme

$$F(G_1+X^mG_2)=F(G_1)+F'(G_1)X^mG_2+F''(G_1)\frac{X^{2m}G_2^2}{2!}+\cdots, \tag{3.6}$$

où G_1 est un polynôme de degré inférieur à m ; le choix de m est ajusté plus loin pour minimiser la complexité. Le premier gain de complexité par rapport à la méthode naïve provient de l'observation suivante.

Lemme 3.16 Étant données les séries G et $F\circ G$, avec $G'(0)$ inversible, la série $F'\circ G$ peut être calculée en $O(\mathsf{M}(N))$ opérations arithmétiques.

Démonstration. La dérivation de séries a une complexité linéaire, et la division est en $O(\mathsf{M}(N))$. Le résultat provient alors de la formule de dérivation d'une composée (déjà utilisée en page 71) :

$$F'\circ G=\frac{(F\circ G)'}{G'}.$$

L'hypothèse sur $G'(0)$ permet de garantir l'inversion du dénominateur. ■

Entrée Deux séries tronquées $F(X) \bmod X^N$ et $G(X) \bmod X^N$ avec $G(0) = 0$ et $G'(0)$ inversible.
Sortie Leur composition $F(G(X)) \bmod X^N$.

1. Soit $m = \lceil \sqrt{N} \rceil$ si la multiplication utilisée emploie la FFT, $m = \lceil \sqrt{N/\log N} \rceil$ sinon.
2. Découper G en $G_1 = G \bmod X^m$ et $G_2 = G - G_1$, poser $H := 1$.
3. Calculer $U = 1/G_1' \bmod X^N$.
4. Calculer $R = S_0 := F(G_1) \bmod X^N$ par la méthode récursive suivante :
   a. Si $\deg F = 1$ renvoyer $F(0) + (F - F(0))G_1 \bmod X^N$,
   b. sinon
      i. soit $k = \lceil \deg(F)/2 \rceil$,
      ii. découper F en $A := F \bmod X^k$ et $B := (F - A)/X^k$,
      iii. calculer récursivement $G_1^{\lceil k/2 \rceil}$, $A(G_1)$, et $B(G_1)$ modulo X^N,
      iv. renvoyer $A(G_1) + G_1^k B(G_1) \bmod X^N$.
5. Pour $i = 1, \ldots, \lceil N/m \rceil$ faire
   a. $H := HG_2 \bmod X^N$,
   b. $S_i := S_{i-1}' U/i \bmod X^N$,
   c. $R := R + S_i H \bmod X^N$.
6. Renvoyer R.

Algorithme 3.8 – Algorithme de composition de Brent–Kung.

L'utilisation répétée de cette idée dans la formule (3.6) mène à une complexité

$$C(F(G_1)) + \frac{N}{m} O(M(N))$$

pour l'ensemble de la composition, où $C(F(G_1))$ représente la complexité du calcul de la première composition avec un *polynôme*, donnée par le lemme suivant.

Lemme 3.17 Soient F et G deux *polynômes* de degrés k et m, avec $G(0) = 0$. Alors, le calcul des N premiers coefficients de la *série* $F \circ G$ peut être effectué en $O(km \log N M(N)/N)$ opérations arithmétiques, ou en $O(km M(N)/N)$ opérations si $\log M(N)/\log N \to \gamma > 1$.

Démonstration du théorème à l'aide de ce lemme. Ce lemme fournit l'estimation

$$C(F(G_1)) = O(m M(N) \log N).$$

La complexité totale est donc bornée par

$$O(m M(N) \log N) + \frac{N}{m} O(M(N)) = O\left(M(N)\left(m \log N + \frac{N}{m}\right)\right).$$

Cette dernière somme est minimisée par le choix de $m = \sqrt{N/\log N}$ qui donne le résultat du théorème. Le cas sans le facteur $\log N$ est laissé en exercice. ■

Démonstration du lemme. Comme d'habitude, on commence par traiter le cas où le degré k est une puissance de 2. Ensuite, quitte à considérer que les coefficients de F pour les degrés allant de $k+1$ jusqu'à la puissance de 2 immédiatement supérieure sont nuls, la perte de complexité est d'au plus un facteur 2, absorbé dans la constante du $O(\cdot)$.

L'idée est d'effectuer ce calcul par « diviser pour régner » en récrivant le polynôme F sous la forme

$$F = F_1 + X^{k/2} F_2,$$

où F_1 et F_2 sont deux polynômes de degré au plus $k/2$. Dans la composition

$$F(G) = F_1(G) + G^{k/2} F_2(G),$$

les deux polynômes du second sommant ont degré au plus $km/2$. La complexité C_k^m découlant de l'utilisation de cette formule vérifie donc

$$C_k^m \leq 2C_{k/2}^m + 2M(\min(N, km/2)) + O(N).$$

C'est dans cette prise de minimum que réside toute la subtilité : tant que k est grand, les calculs sont tronqués à l'ordre N, et dès que k devient suffisamment petit, on exploite l'arithmétique polynomiale. Il vient donc

$$\begin{aligned} C_k^m &\leq 2M(N) + 4M(N) + \cdots + 2^{\ell-1} M(N) + 2^\ell \left(M\left(\frac{km}{2^\ell}\right) + 2M\left(\frac{km}{2^{\ell+1}}\right) + \cdots \right), \\ &\leq 2^\ell M(N) + 2^\ell M\left(\frac{km}{2^\ell}\right) \log\left(\frac{km}{2^\ell}\right), \end{aligned}$$

où la valeur de ℓ, qui conclut la preuve du lemme, est déterminé par

$$\frac{km}{2^\ell} < N \leq \frac{km}{2^{\ell-1}}.$$

■

Exercices

Exercice 3.8 — Exponentielle de série. Soit s une série de terme constant nul et $n = 2^p$. On étudie l'itération définie par $y_0 = z_0 = 1$ et pour $k > 0$

$$\begin{aligned} z_k &= z_{k-1} + z_{k-1}(1 - y_{k-1} z_{k-1}) \bmod X^{2^{k-1}}, \\ y_k &= y_{k-1} - y_{k-1} \int z_k (y'_{k-1} - s' y_{k-1}) \bmod X^{2^k}. \end{aligned}$$

1. Montrer que $y_p - \exp(s) = O(X^n)$;
2. Estimer la complexité du calcul des n premiers coefficients de la série $\exp(s)$ par cette itération ;
3. (Plus difficile) Interpréter cette itération comme l'itération de Newton associée à l'opérateur $Y \mapsto Y' - s'Y$.

■

Exercice 3.9 — Composition avec l'exponentielle. Soit $f(X) \in \mathbb{Q}[[X]]$ une série à coefficients rationnels. Le but de cet exercice est de montrer que les N premiers coefficients de la série $S(X) = f(e^X - 1)$ peuvent être calculés en $O\big(M(N)\log N\big)$ opérations arithmétiques dans $\mathbb{Q}$.

1. Montrer que $S(X) - A(e^X - 1) = O(X^N)$, où $A(X)$ est l'unique polynôme de degré inférieur à N tel que $f(X) = A(X) + O(X^N)$.
2. On pose $B(X) = A(X - 1)$; donner un algorithme pour calculer les coefficients de B à partir de ceux de A en $O\big(M(N)\log N\big)$ opérations dans $\mathbb{Q}$.
3. La transformée de Laplace formelle $\mathcal{L}$ est définie sur $\mathbb{Q}[[X]]$ comme l'application $\mathbb{Q}$-linéaire telle que $\mathcal{L}(X^k) = k!\,X^k$, $k \in \mathbb{N}$. Si $B(X) = b_0 + b_1 X + \cdots + b_{N-1}X^{N-1}$, montrer que
$$\mathcal{L}\big(B(e^X)\big) = \sum_{i=0}^{N-1} \frac{b_i}{1 - iX}.$$
4. Donner un algorithme pour calculer les N premiers coefficients de $B(e^X)$ à partir de ceux de $B(X)$ en $O\big(M(N)\log N\big)$ opérations dans $\mathbb{Q}$.
5. Donner finalement un algorithme calculant les N premiers coefficients de S à partir de ceux de f en $O\big(M(N)\log N\big)$ opérations dans $\mathbb{Q}$.

■

Exercice 3.10 — Composition de séries avec arcsinus. Soit $\mathbb{K}$ un corps de caractéristique nulle, et soit $F \in \mathbb{K}[X]$ une série formelle de terme constant nul. Écrire un algorithme à base d'itérations de Newton permettant de calculer les N premiers termes de la série composée $(\arcsin \circ F)$, à partir des N premiers termes de la série F, en $O(M(N))$ opérations arithmétiques dans $\mathbb{K}$. ■

Exercice 3.11 — Composition de séries formelles en petite caractéristique. Soit $\mathbb{K} = \mathbb{F}_p$ le corps fini à p éléments, où p est un nombre premier. On rappelle que tout élément a de $\mathbb{K}$ vérifie l'égalité $a^p = a$. Le but de cet exercice est de concevoir un algorithme pour la composition des séries dans $\mathbb{K}[[X]]$, de complexité asymptotique quasi-optimale en la précision. Soient f et g deux séries dans $\mathbb{K}[[X]]$ tronquées à précision N, avec $g(0) = 0$, et soit h leur composée $h = f \circ g \bmod X^N$. On suppose $N \geq p$.

1. Montrer que $g(X)^p \bmod X^N$ s'écrit $s(X^p)$ pour une certaine série $s \in \mathbb{K}[[X]]$ connue à précision $\lceil N/p \rceil$.
2. Montrer que le calcul de h se ramène à p compositions de séries à précision $\lceil N/p \rceil$.
3. En déduire un algorithme récursif pour le calcul de h, et estimer sa complexité.

■

Notes

La méthode de Newton remonte à 1669, et se trouve bien décrite dans son traité des fluxions [New40]. Cette méthode a ensuite été raffinée et étendue de nombreuses façons [Yam00 ; Ypm95].

L'utilisation de l'itération de Newton en informatique est très ancienne. L'inversion

de matrice de la Section 3.2 remonte à Schulz [Sch33] pour des matrices réelles. L'importance de la méthode de Newton pour le calcul rapide avec des séries formelles a été soulignée par Lipson [Lip76] : « *In a symbolic mathematics system setting, we feel that Newton's method offers a unified and easily implemented algorithm for solving a wide variety of power series manipulation problems that can be cast into a root-finding mold. Our experience attests to the practicality of Newton's method. We feel that it is a great algebraic algorithm.* »

Du point de vue de la complexité, Sieveking [Sie72] a donné l'Algorithme 3.2 d'inversion rapide de séries. Kung [Kun74] a observé qu'il s'agissait d'une itération de Newton et a amélioré la constante dans la complexité. Ensuite, Brent [Bre76] a montré comment l'exponentielle et le logarithme s'obtiennent également en $O(M(N))$ opérations. Les algorithmes de la Section 3.3 sont dus à Schönhage [Sch82] ; l'application aux sommes et produits composés en Section 3.3 est beaucoup plus récente [Bos+06b]. La méthode de Newton permet également la résolution de systèmes linéaires à coefficients polynomiaux (Chapitre 11) et la résolution d'équations ou de systèmes *différentiels* (Chapitre 13). L'idée d'incorporer le dividende dans la dernière itération de Newton, afin de gagner un facteur constant pour la division de séries (Section 3.2), est due à Karp et Markstein [KM97].

Les algorithmes non triviaux de composition décrits en Section 3.4 sont dus à Brent et Kung [BK78]. Une version optimisée est décrite dans l'article de Johansson [Joh15], qui utilise la formule d'inversion de Lagrange (3.5). L'idée de l'algorithme « pas de bébés, pas de géants » en Section 3.4 remonte à Paterson et Stockmeyer [PS73]. La composition $F \circ G \bmod X^N$ peut s'effectuer en coût quasi-optimal $O(M(N)\log N)$ si G est un polynôme [BK78], ou plus généralement lorsque G est une série algébrique [Hoe02c]. Des compositions avec d'autres fonctions particulières peuvent aussi être calculées en $O(M(N)\log N)$, ou parfois $O(M(N))$ [BSS08]. Savoir si cela est encore possible pour des séries quelconques est un grand problème ouvert. La formule d'inversion évoquée en page 72 a été découverte par Lagrange en 1768 [Lag68].

Dans le cas où la caractéristique p de $\mathbb{A}$ est finie, Bernstein [Ber98] a montré qu'il est possible de descendre la composition à une complexité quasi-optimale en la précision N des séries, pour le prix d'une dépendance linéaire en p. Cet algorithme est intéressant lorsque la caractéristique p est petite. L'exercice 3.11 présente l'idée de l'algorithme dans un cas particulier. Une avancée récente importante est due à Kedlaya et Umans [KU08], qui ont donné le premier algorithme de complexité binaire quasi-linéaire à la fois en N et en $\log p$, pour le calcul de $F \circ G \bmod X^N$, lorsque F et G sont des séries à coefficients dans le corps fini $\mathbb{F}_p$.

Ritzmann [Rit86] a montré que la composition des séries entières peut s'effectuer en complexité *binaire* quasi-optimale : si $F, G \in \mathbb{Z}[X]$ ont des coefficients bornés par ℓ en valeur absolue, alors il est possible de calculer $F \circ G \bmod X^N$ en $O(M_{\mathbb{Z}}(N^2 \log N \log \ell))$ opérations binaires. Il montre aussi que le problème de la composition de séries à coefficients dans un corps $\mathbb{K}$ est d'une difficulté équivalente à celui du calcul du N-ième coefficient de $F \circ G$. Le problème de l'inverse compositionnel leur est également équivalent [BK78].

En présence d'une structure supplémentaire, il est possible de calculer la composition ou l'inverse compositionnel en complexité arithmétique optimale par rapport à la précision. C'est le cas par exemple pour l'inverse compositionnel d'un polynôme,

ou plus généralement d'une série algébrique. C'est également le cas pour la composition $F \circ G$ d'une série différentiellement finie F par une série algébrique G. Dans tous ces cas de figure, la série calculée est différentiellement finie, et les techniques des chapitres sur ces séries (Corollaire 14.10 et Théorème 15.1) s'appliquent : les N premiers termes peuvent se calculer en $O(N)$ opérations arithmétiques, et le N-ième terme en un nombre d'opérations arithmétiques quasi-linéaire en $\sqrt{N}$. Il n'est actuellement pas connu si une complexité linéaire en N peut être obtenue pour d'autres types de structures, comme par exemple pour le calcul de l'inverse compositionnel $F^{(-1)}(X) \bmod X^N$ d'une série différentiellement finie F, ou de la composition $F \circ G \bmod X^N$ de deux séries différentiellement finies F et G.

En pratique, les constantes cachées dans les $O(\cdot)$ jouent un grand rôle, et nécessitent généralement l'implantation à la fois des algorithmes asymptotiquement meilleurs et des algorithmes plus classiques, souvent plus efficaces en petite taille. Dans le modèle de multiplication polynomiale par FFT, les meilleurs constantes devant $M(N)$ sont dues à Harvey [Har10 ; Har11] : $\frac{13}{9}$ pour l'inverse, $\frac{4}{3}$ pour la racine carrée, $\frac{13}{6}$ pour l'exponentielle. Ces problèmes ont aussi été étudiés par van der Hoeven [Hoe10]. Des articles de synthèse dus à Bernstein [Berb ; Ber08] décrivent diverses techniques utilisées pour éliminer certains calculs redondants dans la méthode de Newton.

L'Exercice 3.8 est inspiré par [HZ], et l'Exercice 3.9 est tiré de [BSS08].

D'autres modèles, appelés *paresseux* et *détendus*, permettent de manipuler les séries de façon plus naturelle sans se soucier de la précision, qui est gérée de façon automatique. Chaque série est vue comme le vecteur des coefficients déjà calculés et une fonction permettant de calculer un coefficient de degré N en fonction des précédents [BHL11 ; Hoe02c ; Hoe03 ; Hoe07].

Bibliographie

Berb — Bernstein, Daniel J. (2001). *Removing redundancy in high-precision Newton iteration*. URL : http://cr.yp.to/fastnewton.html (visible en 2001).

Ber08 — — (2008). « Fast multiplication and its applications ». In : *Algorithmic number theory : lattices, number fields, curves and cryptography*. Vol. 44. Publications of the Research Institute for Mathematical Sciences. Cambridge University Press, p. 325–384.

Ber98 — — (1998). « Composing power series over a finite ring in essentially linear time ». In : *Journal of Symbolic Computation*, vol. 26, n°3, p. 339–341.

BHL11 — Berthomieu, Jérémy, Joris van der Hoeven et Grégoire Lecerf (2011). « Relaxed algorithms for *p*-adic numbers ». In : *Journal de Théorie des Nombres de Bordeaux*, vol. 23, n°3, p. 541–577.

BK78 — Brent, R. P. et H. T. Kung (1978). « Fast algorithms for manipulating formal power series ». In : *Journal of the ACM*, vol. 25, n°4, p. 581–595.

Bos+06b — Bostan, Alin, Philippe Flajolet, Bruno Salvy et Éric Schost (2006). « Fast computation of special resultants ». In : *Journal of Symbolic Computation*, vol. 41, n°1, p. 1–29.

Bre76 Brent, Richard P. (1976). « Multiple-precision zero-finding methods and the complexity of elementary function evaluation ». In : *Analytic computational complexity*. Proceedings of a Symposium held at Carnegie-Mellon University, Pittsburgh, PA, 1975. Academic Press, p. 151–176.

BSS08 Bostan, Alin, Bruno Salvy et Éric Schost (2008). « Power series composition and change of basis ». In : *ISSAC'08 : International Symposium on Symbolic and Algebraic Computation*. Éd. par David J. Jeffrey. ACM Press, p. 269–276.

Har10 Harvey, David (2010). « Faster exponentials of power series ». url : https://arxiv.org/abs/0911.3110.

Har11 — (2011). « Faster algorithms for the square root and reciprocal of power series ». In : *Mathematics of Computation*, vol. 80, p. 387–394.

Hoe02c Hoeven, Joris van der (2002). « Relax, but don't be too lazy ». In : *Journal of Symbolic Computation*, vol. 34, n°6, p. 479–542.

Hoe03 — (2003). « Relaxed multiplication using the middle product ». In : *ISSAC'03 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Manuel Bronstein. Philadelphia, USA : ACM Press, p. 143–147.

Hoe07 — (2007). « New algorithms for relaxed multiplication ». In : *Journal of Symbolic Computation*, vol. 42, n°8, p. 792–802.

Hoe10 — (2010). « Newton's method and FFT trading ». In : *Journal of Symbolic Computation*, vol. 45, n°8, p. 857–878.

HZ Hanrot, Guillaume et Paul Zimmermann (2004). *Newton iteration revisited*. url : http://www.loria.fr/~zimmerma/papers (visible en 2004).

Joh15 Johansson, Fredrik (2015). « A fast algorithm for reversion of power series ». In : *Mathematics of Computation*, vol. 84, n°291, p. 475–484.

KM97 Karp, A. H. et P. Markstein (1997). « High-precision division and square root ». In : *ACM Transactions on Mathematical Software*, vol. 23, n°4, p. 561–589.

KU08 Kedlaya, Kiran S. et Christopher Umans (2008). « Fast modular composition in any characteristic ». In : *FOCS'08 : IEEE Conference on Foundations of Computer Science*. Washington, DC, USA : IEEE Computer Society, p. 146–155.

Kun74 Kung, H. T. (1974). « On computing reciprocals of power series ». In : *Numerische Mathematik*, vol. 22, p. 341–348.

Lag68 Lagrange, Joseph Louis, comte de (1768). « Nouvelle méthode pour résoudre les équations littérales par le moyen des séries ». In : *Mémoires de l'Académie Royale des Sciences et Belles-Lettres de Berlin*, vol. 24, p. 251–326.

Lip76 Lipson, J. D. (1976). « Newton's method : a great *algebraic* algorithm ». In : *SYMSAC'76 : ACM Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 260–270.

New40 Newton, Isaac (1740). *La méthode des fluxions, et les suites infinies*. Traduction française par G. Buffon du texte de 1671 de Newton. de Bure. url : http://www.gallica.fr.

PS73 Paterson, M. S. et L. J. Stockmeyer (1973). « On the number of nonscalar multiplications necessary to evaluate polynomials ». In : *SIAM Journal on Computing*, vol. 2, n°1, p. 60–66.

Rit86 Ritzmann, P. (1986). « A fast numerical algorithm for the composition of power series with complex coefficients ». In : *Theoretical Computer Science*, vol. 44, p. 1–16.

Sch33 Schulz, G. (1933). « Iterative Berechnung der reziproken Matrix ». In : *Zeitschrift für angewandte Mathematik und Physik*, vol. 13, p. 57–59.

Sch82 Schönhage, A. (1982). *The fundamental theorem of algebra in terms of computational complexity*. Preliminary Report. 73 pages. Tübingen, Germany : Mathematisches Institut der Universität.

Sie72 Sieveking, M. (1972). « An algorithm for division of powerseries ». In : *Computing*, vol. 10, p. 153–156.

Yam00 Yamamoto, Tetsuro (2000). « Historical developments in convergence analysis for Newton's and Newton-like methods ». In : *Journal of Computational and Applied Mathematics*, vol. 124, n°1-2, p. 1–23.

Ypm95 Ypma, Tjalling J. (1995). « Historical development of the Newton–Raphson method ». In : *SIAM Review*, vol. 37, n°4, p. 531–551.

4. Division euclidienne, fractions rationnelles et récurrences linéaires à coefficients constants

Résumé

La multiplication et l'inversion rapide de séries permettent le calcul efficace de la division euclidienne de polynômes et du développement en série des fractions rationnelles. Ils mènent en particulier au calcul rapide d'un ou de plusieurs termes d'une suite récurrente linéaire à coefficients constants.

4.1 Introduction

Les suites récurrentes linéaires à coefficients constants sont extrêmement présentes dans la nature. On estime ainsi qu'environ 12% des suites de la littérature [1] sont de ce type. L'exemple le plus célèbre est celui de la suite de Fibonacci, définie par $F_0 = 0$, $F_1 = 1$ et

$$F_{n+2} = F_{n+1} + F_n, \quad n \geq 0. \tag{4.1}$$

L'utilisation directe de cette récurrence permet de calculer les N premiers éléments de la suite de manière optimale du point de vue de la complexité. En revanche, pour le calcul d'un seul terme d'indice élevé, le passage par le calcul de tous ses prédécesseurs est inutilement coûteux. Une première idée de ce chapitre consiste à utiliser une version matricielle des récurrences pour aboutir à une complexité quasi-optimale pour cette question.

La récurrence (4.1) n'est que d'ordre 2. Pour des suites définies par des récurrences d'ordre d plus grand, il faut aussi chercher à limiter le coût des calculs vis-à-vis de d. Simplement dérouler la récurrence n'est plus alors la meilleure façon de calculer les N premiers termes de la suite. Une autre idée importante de ce chapitre est d'exploiter le lien entre les récurrences linéaires à coefficients constants et les fractions

1. Recensées très largement par l'encyclopédie en ligne des suites d'entiers, http://oeis.org.

rationnelles, qui mène à une complexité quasi-optimale vis-à-vis à la fois de N et d pour cette question, ainsi que pour celle du calcul des N premiers termes, et aussi pour le développement en série des fractions rationnelles.

Les applications algorithmiques de ces idées sont nombreuses. Certaines d'entre elles sont présentées dans des exercices de ce chapitre, comme le calcul rapide d'un polynôme sur $1, 2, \ldots, N$, une partie du calcul du polynôme minimal d'une matrice, et des tests probabilistes de primalité d'entiers.

4.2 Division de polynômes

Étant donnés deux polynômes F et G de $\mathbb{A}[X]$, avec G unitaire, à coefficients dans un anneau $\mathbb{A}$, il existe d'uniques polynômes Q et R, quotient et reste de la division euclidienne de F par G, tels que :

$$F = QG + R \quad \text{avec} \quad \deg R < \deg G.$$

Calculer rapidement le quotient Q et le reste R est d'importance vitale dans toute la suite de cet ouvrage. Outre l'application aux suites récurrentes qui sera détaillée plus loin, les algorithmes de division sont utilisés au Chapitre 5 pour l'évaluation multipoint et l'interpolation et au Chapitre 6 pour le calcul de pgcd. À leur tour, ces deux algorithmes sont centraux dans nombre d'applications.

Méthode naïve

L'algorithme naïf pour calculer Q et R consiste à poser la division. Pour cela, les termes dominants sont d'abord isolés :

$$F = aX^m + T_F, \quad G = X^n + T_G, \qquad \text{avec} \quad \deg T_F < m, \quad \deg T_G < n.$$

Si $m < n$, il n'y a rien à faire. Sinon, la boucle élémentaire de la division de F par G consiste à « tuer » le terme de plus haut degré de F en effectuant la soustraction

$$F - aX^\delta G = (aX^m + T_F) - \left(aX^{\delta+n} + aX^\delta T_G\right) = T_F - aX^\delta T_G,$$

où $\delta = m - n$. Le polynôme obtenu est congru à F modulo G, mais de degré strictement inférieur à m. Il n'y a plus qu'à itérer ce procédé jusqu'à obtenir un polynôme de degré strictement inférieur à n, et à garder trace des quotients successifs.

La complexité de cet algorithme est facile à estimer. La boucle élémentaire présentée ci-dessus s'effectue en $O(n)$ opérations de type $(+, -, \times)$; dans le pire des cas, l'algorithme effectue $(m - n + 1)$ passages dans cette boucle, de sorte que la complexité de la division de F par G est de $O(n(m-n))$ opérations. Le pire des cas est atteint pour $m = 2n$. C'est donc un algorithme *quadratique* (mais un bon algorithme quadratique : la constante dans le $O(\cdot)$ est en fait petite, c'est 2). Le même genre d'estimation s'obtient pour la division euclidienne classique des entiers.

Algorithme rapide

Un bien meilleur résultat peut être obtenu à base d'itération de Newton. L'idée principale de l'algorithme part de la récriture de $F = QG + R$ en

$$\frac{F}{G} = Q + \frac{R}{G}.$$

Entrée F, G de degrés m et n.
Sortie Q et R tels que $F = QG + R$ avec $\deg R < \deg G$.

1. Calculer $T^m F(1/T)$ et $T^n G(1/T)$ (aucune opération arithmétique n'est nécessaire, il suffit d'inverser l'ordre des coefficients).
2. Calculer le quotient $(T^m F(1/T))/(T^n G(1/T)) \bmod T^{m-n+1}$ par une inversion de série formelle suivie d'un produit.
3. En déduire Q en inversant l'ordre des coefficients.
4. En déduire R, qui est donné par $R = F - QG \bmod X^n$.

Algorithme 4.1 – Division euclidienne rapide.

Si on pense que $\mathbb{A} = \mathbb{R}$, on voit donc que le développement asymptotique de F/G *à l'infini* a ses premiers termes donnés par Q, puisque R/G tend vers 0 à l'infini (à cause des contraintes de degré sur R et G). Ceci suggère que l'on va obtenir Q par calcul du développement de Taylor de F/G au voisinage de l'infini.

Concrètement, il suffit de ramener d'abord l'infini en zéro (par le changement de variable $X = 1/T$), pour réduire le calcul à la division de séries formelles. Plus précisément, le point de départ est l'identité

$$\frac{T^m F(1/T)}{T^n G(1/T)} = T^{m-n} Q(1/T) + \frac{T^m R(1/T)}{T^n G(1/T)}.$$

Dans cette identité les numérateurs et dénominateurs des fractions sont des polynômes, et le polynôme $T^n G(1/T)$ a 1 pour terme constant. En outre, la valuation du second sommant du membre droit est supérieure à $m - n$. En découle donc l'Algorithme 4.1 de division rapide.

Théorème 4.1 Soient G un polynôme unitaire de $\mathbb{A}[X]$ de degré n et $F \in \mathbb{A}[X]$ de degré $m \geq n$. Alors on peut calculer le quotient Q et le reste R de la division euclidienne de F par G en $5M(m-n) + M(n) + O(m)$ opérations $(+, -, \times)$ de $\mathbb{A}$.

Ainsi, la division euclidienne ne coûte pas plus cher que la multiplication (à une constante près). En particulier, si on utilise une multiplication à base de FFT, le coût de la division est *linéaire* en le degré, à des facteurs logarithmiques près.

Démonstration. D'après la Proposition 3.4 et la remarque qui la suit, l'inverse du dénominateur à l'étape (2) s'obtient en

$$4M(m-n) + O(m-n)$$

opérations dans $\mathbb{A}$, puis une multiplication supplémentaire donne Q. Pour retrouver R, le dernier produit est effectué modulo X^n, c'est-à-dire pour un coût de $M(n)$. Toutes les additions sont prises en compte dans le terme $O(m)$. ■

R Si Q a un degré beaucoup plus petit que G, on peut accélérer le calcul de QG, en découpant G en « tranches » de taille $m-n$; de la sorte, le dernier produit peut se faire en $\frac{n}{m-n}\mathsf{M}(m-n)$ opérations. Enfin, si de nombreuses réductions sont à faire modulo le même polynôme G, il est évidemment recommandé de stocker l'inverse du réciproque $\mathrm{T}^n\mathrm{G}(1/\mathrm{T})$ de G.

Le cas des entiers

Comme pour la multiplication, il est possible de poser le problème dans $\mathbb{Z}$ comme dans un anneau de polynômes. C'est ce dernier cas qui est le plus simple à étudier, puisqu'il n'y a pas à y gérer de retenue. On obtient cependant un résultat analogue sur les entiers.

Théorème 4.2 Soit $\mathsf{M}_{\mathbb{Z}}$ une fonction de multiplication pour $\mathbb{Z}$, et f et g deux entiers positifs avec $g \leq f \leq 2^n$. Soient q et r les quotient et reste de la division euclidienne de f par g :

$$f = qg + r \quad \text{avec} \quad 0 \leq r < g.$$

On peut calculer q et r en $\mathrm{O}(\mathsf{M}_{\mathbb{Z}}(n))$ opérations binaires.

Application aux calculs modulaires

Une application importante de la division euclidienne rapide est le calcul efficace dans des structures algébriques de type « quotient », par exemple dans n'importe quel corps fini.

Corollaire 4.3 Si $\mathbb{A}$ est un anneau et si P est un polynôme unitaire de degré n de $\mathbb{A}[\mathrm{X}]$, alors dans $\mathbb{A}[\mathrm{X}]/(\mathrm{P})$, l'addition et la multiplication par un élément de $\mathbb{A}$ ont une complexité linéaire en n, la multiplication a une complexité en $\mathrm{O}(\mathsf{M}(n))$.

Le calcul de l'inverse (des éléments inversibles) dans $\mathbb{A}[\mathrm{X}]/(\mathrm{P})$ est un peu plus complexe. Un algorithme de complexité $\mathrm{O}(\mathsf{M}(n)\log n)$ est présenté au Chapitre 6.

Démonstration. Il s'agit de calculer modulo P. L'addition et la multiplication par un scalaire s'effectuent terme à terme, ainsi leur complexité est linéaire en le degré n de P. Ensuite, pour multiplier deux éléments $\mathrm{A}+(\mathrm{P})$ et $\mathrm{B}+(\mathrm{P})$ de $\mathbb{A}[\mathrm{X}]/(\mathrm{P})$, on commence par multiplier A et B dans $\mathbb{A}[\mathrm{X}]$, puis on réduit le résultat modulo P. La complexité découle donc de ce qui précède. ■

La question du calcul dans $\mathbb{Z}/n\mathbb{Z}$ est légèrement plus délicate que son analogue dans $\mathbb{A}[\mathrm{X}]$, à cause des retenues. Malgré tout, les résultats s'étendent.

4.3 Suites récurrentes linéaires à coefficients constants

Définition 4.1 Une suite $(a_n)_{n\geq 0}$ d'éléments de l'anneau $\mathbb{A}$ est appelée suite récurrente linéaire à coefficients constants (srlcc) d'ordre d si elle satisfait une récurrence

de la forme

$$a_{n+d} = p_{d-1}a_{n+d-1} + \cdots + p_0 a_n, \qquad n \geq 0,$$

où les p_i sont des éléments de $\mathbb{A}$. Le polynôme $\mathrm{P} = \mathrm{X}^d - p_{d-1}\mathrm{X}^{d-1} - \cdots - p_0$ de $\mathbb{A}[\mathrm{X}]$ est appelé *polynôme caractéristique* de la suite $(a_n)_{n\geq 0}$.

Il faut noter que tout multiple d'un polynôme caractéristique est aussi caractéristique. Le pgcd de ces polynômes caractéristiques est appelé polynôme *minimal* de la suite.

Exemples

Exercice 4.1 Soit M une matrice de taille $d \times d$ à coefficients dans $\mathbb{A}$, de polynôme caractéristique $\chi_{\mathrm{M}}(\mathrm{X}) = \det(\mathrm{X}\mathrm{I}_d - \mathrm{M})$, soient u une matrice ligne et v une matrice colonne. Montrer que la suite $(u\mathrm{M}^n v)_{n\geq 0}$ est une suite récurrente linéaire à coefficients constants admettant χ_{M} comme polynôme caractéristique. ∎

Exercice 4.2 Soit $a_1, \ldots, a_r$ des entiers strictement positifs. Soit A_n et B_n le nombre de r-uplets non ordonnés, resp. ordonnés, $(x_1, \ldots, x_r) \in \mathbb{N}^r$, solutions de l'équation $a_1 x_1 + \cdots + a_r x_r = n$. Montrer que les suites (A_n) et (B_n) sont récurrentes linéaires à coefficients constants, admettant $(\mathrm{X}^{a_1} - 1)\cdots(\mathrm{X}^{a_r} - 1)$ et $\mathrm{X}^{a_1} + \cdots + \mathrm{X}^{a_r} - 1$ comme polynômes caractéristiques. ∎

Exercice 4.3 Si $\mathrm{P} \in \mathbb{A}[\mathrm{X}]$ est de degré d, alors la suite $(\mathrm{P}(n))_{n\geq 0}$ est une srlcc de polynôme caractéristique $(\mathrm{X} - 1)^{d+1}$. ∎

Méthode naïve

L'approche naïve pour le calcul des N premiers termes d'une srlcc consiste tout simplement à dérouler la récurrence et requiert $\mathrm{O}(d\mathrm{N})$ opérations dans $\mathbb{A}$.

Exponentiation binaire

Pour calculer uniquement le N-ième terme de la suite, une alternative à cette méthode naïve est basée sur l'exponentiation binaire de la matrice compagnon associée à la suite. Par exemple, la récurrence (4.1) se récrit matriciellement

$$\begin{pmatrix} \mathrm{F}_{\mathrm{N}} \\ \mathrm{F}_{\mathrm{N}+1} \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}}_{\mathrm{C}} \begin{pmatrix} \mathrm{F}_{\mathrm{N}-1} \\ \mathrm{F}_{\mathrm{N}} \end{pmatrix} = \mathrm{C}^{\mathrm{N}} \begin{pmatrix} \mathrm{F}_0 \\ \mathrm{F}_1 \end{pmatrix}, \qquad \mathrm{N} \geq 1.$$

La puissance de la matrice constante C se calcule alors récursivement par

$$\mathrm{C}^{\mathrm{N}} = \begin{cases} (\mathrm{C}^{\mathrm{N}/2})^2, & \text{si N est pair,} \\ \mathrm{C} \cdot (\mathrm{C}^{\frac{\mathrm{N}-1}{2}})^2, & \text{sinon.} \end{cases}$$

On en déduit que le calcul de F_{N} peut être effectué sans calculer les précédents en $\mathrm{O}(\log \mathrm{N})$ produits de matrices 2×2, soit $\mathrm{O}(\log \mathrm{N})$ opérations dans $\mathbb{A}$.

Cette idée s'étend facilement à des récurrences d'ordre arbitraire.

Exercice 4.4 Montrer que le N-ième terme de toute récurrence linéaire d'ordre d à coefficients constants peut se calculer en $O(d^3 \log N)$ opérations dans $\mathbb{A}$. ■

L'utilisation d'une multiplication matricielle rapide abaisse cette complexité vis-à-vis de d à $O(\mathsf{MM}(d)\log N) = O(d^\theta \log N)$ opérations dans $\mathbb{A}$ (voir le Chapitre 8). La dépendance en d reste cependant plus que quadratique. La section suivante présente une meilleure méthode, de complexité quasi-linéaire en d.

Exponentiation modulaire

La complexité arithmétique de l'algorithme de la section précédente (Exercice 4.4) est quasi-optimale par rapport à l'indice N, mais n'est pas très bonne par rapport à l'ordre d de la récurrence.

Une méthode plus efficace est obtenue en exploitant mieux la structure du problème. En effet, les matrices multipliées ne sont pas quelconques, mais sont des puissances d'une matrice compagnon. Cette structure peut être exploitée grâce à l'observation suivante.

Lemme 4.4 Soit $(a_n)_{n\geq 0}$ une suite récurrente linéaire à coefficients constants de polynôme caractéristique $P(X)$. Soit $\mathbb{B} = \mathbb{A}[X]/(P)$ et x la classe de X dans $\mathbb{B}$. Alors la matrice compagnon C de P est à la fois

1. telle que $(a_{n+1},\dots,a_{n+d}) = (a_n,\dots,a_{n+d-1})\cdot C$ pour tout $n \geq 0$;
2. matrice de l'application $\mathbb{A}$-linéaire de multiplication par x dans $\mathbb{B}$ dans la base canonique $\{1, x, \dots, x^{d-1}\}$.

Démonstration. Il suffit d'écrire la matrice pour le voir. Si $P = X^d - p_{d-1}X^{d-1} - \cdots - p_0$, la matrice C vaut par définition

$$C = \begin{pmatrix} 0 & 0 & \dots & 0 & p_0 \\ 1 & 0 & \dots & 0 & p_1 \\ 0 & 1 & \dots & 0 & p_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & p_{d-1} \end{pmatrix}.$$

La première propriété traduit la définition de la récurrence :

$$a_{n+d} = p_{d-1}a_{n+d-1} + \cdots + p_0 a_n.$$

La seconde exprime que modulo P, $X^d = p_{d-1}X^{d-1} + \cdots + p_0$. ■

Corollaire 4.5 Avec les mêmes notations, pour tous entiers positifs i et k,

$$a_{k+i} = (a_i,\dots,a_{d+i-1})\cdot V_k,$$

où V_k a pour coordonnées les coefficients du polynôme X^k mod P.

Démonstration. Le vecteur V_k est la première colonne de C^k, et d'après le lemme précédent, cette matrice est celle de la multiplication par X^k dans $\mathbb{B}$. Les coefficients de la première colonne sont donc les coordonnées de $x^k \cdot 1$ dans la base $\{1, x, \dots, x^{d-1}\}$. ■

Entrée $(p_0,\dots,p_{d-1})$ les coefficients d'une récurrence linéaire ; $(a_0,\dots,a_{d-1})$ des conditions initiales ; un entier N.
Sortie Le terme a_N de la suite définie par $a_{n+d} = p_{d-1}a_{n+d-1} + \cdots + p_0 a_n$.

1. Poser $P = X^d - p_{d-1}X^{d-1} - \cdots - p_0$ et $x = X \bmod P$.
2. Calculer récursivement $Q = x^N = q_0 + q_1 x + \cdots + q_{d-1}x^{d-1}$ par

$$x^k = \begin{cases} (x^{k/2})^2, & \text{si } k \text{ est pair,} \\ x \cdot (x^{\frac{k-1}{2}})^2, & \text{sinon.} \end{cases}$$

3. Renvoyer $q_0 a_0 + \cdots + q_{d-1}a_{d-1}$.

Algorithme 4.2 – Calcul du N-ième terme d'une srlcc.

Le résultat de complexité est alors le suivant.

Théorème 4.6 Soit (a_n) une suite récurrente linéaire à coefficients constants, donnée par une récurrence d'ordre d et des conditions initiales $a_0,\dots,a_{d-1}$. Soit $N \geq d$. Le calcul du N-ième terme a_N peut se faire en seulement $O(M(d)\log N)$ opérations dans $\mathbb{A}$.

Démonstration. La méthode est présentée dans l'Algorithme 4.2. Chaque étape du calcul récursif a lieu dans $\mathbb{A}[X]/(P)$ et coûte $O(M(d))$ opérations dans $\mathbb{A}$ d'après le Corollaire 4.3, le nombre d'étapes est en $\log N$; la dernière multiplication ne prend que $O(d)$ opérations. La complexité est donc $O(M(d)\log N)$. ■

4.4 Développement de fractions rationnelles

Il est également possible de calculer efficacement les N premiers éléments d'une suite récurrente linéaire à coefficients constants, ou de manière équivalente, le développement de Taylor des fractions rationnelles.

Séries rationnelles et suites récurrentes

Ces deux notions sont deux aspects de la même question.

Lemme 4.7 Soit $A(X) = \sum_{n\geq 0} a_n X^n$ la série génératrice de la suite $(a_n)_{n\geq 0}$. Les assertions suivantes sont équivalentes :

i. La suite (a_n) est une srlcc, ayant P pour polynôme caractéristique, de degré d.
ii. $A(X) = N_0/\bar{P}$ où $\bar{P} = P(1/X)X^d$, pour un certain $N_0 \in \mathbb{K}[X]$ avec $\deg(N_0) < d$.

De plus, si P est le polynôme minimal de (a_n), alors $d = \max\{1 + \deg(N_0), \deg(\bar{P})\}$ et $\operatorname{pgcd}(N_0, \bar{P}) = 1$.

Démonstration. Supposons que P s'écrive $P = g_d X^d + \cdots + g_0$. Pour l'équivalence, on utilise uniquement le fait que le coefficient de X^{d+i} dans $\bar{P}\cdot A(X)$ est égal à $g_d a_{i+d} +$

$\cdots + g_0 a_i$ et que $\bar{P}(0) = g_d \neq 0$.

Soit maintenant P le polynôme minimal de (a_n). On a $\deg(\bar{P}) \leq d$ avec égalité si et seulement si $g_0 \neq 0$, c'est-à-dire $X \nmid P$. Donc $d \geq \max\{1 + \deg(N_0), \deg(\bar{P})\}$. Supposons par l'absurde que cette inégalité est stricte. Alors $X \mid P$ et on a que P/X est aussi polynôme caractéristique de (a_n), ce qui contredit la minimalité de P. Donc $d = \max\{1 + \deg(N_0), \deg(\bar{P})\}$.

Soit enfin $u := \mathrm{pgcd}(N_0, \bar{P})$. Alors $P_u := P/\bar{u}$ est un polynôme de degré $d - \deg(u)$ qui est caractéristique de (a_n), car $\bar{P}/u = \bar{P}_u$ et $(\bar{P}/u) \cdot A(X) = N_0/u$ est un polynôme de degré $< d - \deg(u)$. Par la minimalité, cela implique que $\deg(u) = 0$, donc N_0 et $\bar{P}$ sont bien premiers entre eux. ■

Cette équivalence et le Théorème 4.6 ont pour conséquence immédiate un résultat de complexité.

Corollaire 4.8 Soit $F(X)/G(X)$ dans $\mathbb{A}(X)$ de degré au plus d, avec $G(0)$ inversible. Le N-ième coefficient du développement en série

$$F(X)/G(X) = \sum_{n \geq 0} a_n X^n$$

peut être calculé en $O(M(d) \log N)$ opérations.

Développement

On peut calculer le développement de Taylor à l'ordre N d'une fraction rationnelle de degré $d \leq N$ en $O(M(N))$ opérations, en utilisant la méthode de Newton, décrite au Chapitre 3. Mais $O(M(N))$ est en général beaucoup plus gros que les $O(dN)$ opérations requises par la méthode naïve. Une complexité linéaire en N tout en ne croissant pas trop par rapport à d est en fait possible.

Théorème 4.9 Soit $F(X)/G(X)$ dans $\mathbb{A}(X)$ avec numérateur et dénominateur de degrés au plus d, et $G(0)$ inversible. Le développement de Taylor de $F(X)/G(X)$ à précision $N \geq d$ peut se calculer en $O(NM(d)/d)$ opérations d'anneau $(+, -, \times)$ dans $\mathbb{A}$.

Si l'anneau $\mathbb{A}$ permet la FFT, cette estimation de complexité devient $O(N \log d)$ ce qui est quasi-optimal, simultanément vis-à-vis de N et de d.

Le point de départ est le Corollaire 4.5 qui montre comment calculer un coefficient de la suite en fonction de coefficients distants. L'idée est de calculer les coefficients par tranches de longueur d. Il ne suffit pas d'utiliser d fois le corollaire, mais il faut encore une fois exploiter une structure supplémentaire, qui provient ici de ce que les coefficients cherchés sont consécutifs. Cette idée est résumée par le résultat suivant.

Lemme 4.10 Soit $F(X)/G(X)$ dans $\mathbb{A}(X)$ avec $G(0)$ inversible. Soit d le degré de G et soit $k \geq 0$. Alors les coefficients $a_k, a_{k+1}, \ldots, a_{k+d-1}$ du développement de

Entrée Deux polynômes $F(X), G(X)$ avec
$\deg G = d$, $G(0) \neq 0$ et $\deg F < d$, un entier $N > d$.
Sortie Les coefficients $a_0, \ldots, a_N$ du développement
$F(X)/G(X) = \sum_{n \geq 0} a_n X^n$.

1. Calculer les $2d-1$ premiers coefficients $a_0, \ldots, a_{2d-2}$.
2. Poser $y_1 = X^{d-1} \bmod \bar{G}(X)$ et $y^\star = X^d \bmod \bar{G}(X)$,
où $\bar{G}(X) = X^d G(1/X)$.
3. Pour $i = 2, \ldots, \lceil N/d \rceil$:
   a. Calculer $y_i = y^\star y_{i-1} \bmod \bar{G}(X)$.
   b. Calculer $P = \left(a_{2d-2} + \cdots + a_0 X^{2d-2}\right) y_i$.
   c. Extraire $a_{id-1}, a_{id}, \ldots, a_{(i+1)d-2}$ les coefficients des monômes $X^{2d-2}, \ldots, X^d, X^{d-1}$ dans P.
4. Renvoyer $a_0, \ldots, a_N$.

Algorithme 4.3 – Développement en série d'une fraction rationnelle.

$F(X)/G(X) = \sum_i a_i X^i$ sont les coefficients de $X^{2d-2}, \ldots, X^d, X^{d-1}$ du produit

$$\left(a_{2d-2} + \cdots + a_0 X^{2d-2}\right)\left(X^k \bmod G(1/X)X^d\right).$$

Démonstration. Le polynôme $G(1/X)X^d$ est polynôme caractéristique de la suite des coefficients d'après le Lemme 4.7. Le coefficient de X^{k+i} pour $i = 0, \ldots, d-1$ dans le produit ci-dessus est égal au produit scalaire $(A_i \mid V_k)$ où $A_i = (a_i, \ldots, a_{d+i-1})$, d'après le Corollaire 4.5. ■

Preuve du Théorème 4.9. La méthode est résumée dans l'Algorithme 4.3. La première étape se calcule par itération de Newton en $O(M(d))$ opérations dans $\mathbb{A}$, ce qui représente un coût négligeable sur l'ensemble. L'obtention de $y^\star$ est une récriture à partir de $\bar{G}$ et ne demande pas d'opération. Ensuite, chaque itération utilise un produit dans $\mathbb{A}[X]/(\bar{G})$ et un produit d'un polynôme de degré $2d-2$ par un polynôme de degré $d-1$. Au total, sont donc effectuées $O(NM(d)/d)$ opérations dans $\mathbb{A}$. ■

La traduction de ce résultat au niveau des srlcc est immédiate.

Corollaire 4.11 Soit (a_n) une suite récurrente linéaire à coefficients constants, donnée par une récurrence d'ordre d, et des conditions initiales $a_0, \ldots, a_{d-1}$. Soit $N \geq d$. Alors, les termes $a_0, \ldots, a_N$ peuvent être calculés en $O(NM(d)/d)$ opérations dans $\mathbb{A}$.

4.5 Applications

Évaluation d'un polynôme sur une progression arithmétique

Corollaire 4.12 Un polynôme P de degré d peut être évalué aux $N+1 \gg d$ points $b, a+b, 2a+b, \ldots, Na+b$ au prix total de $O(NM(d)/d)$ opérations arithmétiques.

Démonstration. L'opérateur $\Delta_a(\mathrm{P}) = \mathrm{P}(\mathrm{X}+a) - \mathrm{P}(\mathrm{X})$ fait décroître le degré des polynômes. Il s'ensuit que $\Delta_a^{d+1}\mathrm{P} = 0$ et donc $\mathrm{P}(an+b)$ est une suite récurrente linéaire d'ordre $d+1$ et de polynôme caractéristique $(\mathrm{X}-1)^{d+1}$. ■

Propriétés de clôture

La classe des srlcc admet de nombreuses propriétés de clôture : si $\mathbf{a} = (a_n)_n$ et $\mathbf{b} = (b_n)_n$ sont deux srlcc de polynômes caractéristiques P et Q, alors

1. la somme $\mathbf{a} + \mathbf{b} = (a_n + b_n)_n$ et le produit de Cauchy $\mathbf{a} \star_{\mathbf{C}} \mathbf{b}$, dont le terme général est $\sum_{i=0}^{n} a_i b_{n-i}$, sont deux srlcc de polynôme caractéristique PQ ;
2. le produit d'Hadamard $\mathbf{a} \odot \mathbf{b} = (a_n b_n)_n$ est une srlcc de polynôme caractéristique égal au produit composé $\mathrm{P} \otimes \mathrm{Q}$ défini au Chapitre 3 ;
3. la suite $\left(\sum_{i=0}^{n} \binom{n}{i} a_i b_{n-i}\right)_n$ est une srlcc de polynôme caractéristique égal à la somme composée $\mathrm{P} \oplus \mathrm{Q}$ définie au Chapitre 3.

Ils se calculent donc tous en bonne complexité (quasi-linéaire en la taille de la sortie).

Exercice 4.5 Prouver les assertions précédentes. ■

Tests de primalité

Une application du calcul rapide d'un terme d'une récurrence est une famille de tests probabilistes de primalité, de complexité polynomiale. L'idée est de construire une suite récurrente (a_n) d'entiers telle que la primalité de n soit équivalente (ou presque équivalente) à $a_n \equiv 0 \bmod n$.

Un cas particulier important en est *le test de Fermat* (pour lequel $a_n = a^{n-1} - 1$) implanté dans la plupart des systèmes de calcul formel. Bien qu'il soit probabiliste (si n ne passe pas le test, n est composé, mais si n passe le test, alors il est premier seulement avec une grande probabilité), sa grande simplicité le rend souvent préférable à d'autres algorithmes sophistiqués.

Exercice 4.6 Soit (a_n) une srlcc d'ordre d. Montrer qu'il existe des constantes entières $c_0, c_1, \ldots, c_d$ telles que p divise $c_0 + c_1 a_{p-1} + \cdots + c_d a_{p-d}$ dès lors que p est un nombre premier. De plus, pour tout premier p, les constantes $c_i \bmod p$ peuvent être trouvées en $\mathrm{O}(\mathsf{M}(d))$ opérations arithmétiques dans $\mathbb{A} = \mathbb{Z}/p\mathbb{Z}$. [Indication : pour A une matrice carrée d'entiers et p un nombre premier, la trace de A^p et celle de A sont congrues modulo p.] ■

Par exemple, si (F_n) est la suite de Fibonacci, alors p divise $\mathrm{F}_{p-2} + 3\mathrm{F}_{p-1} - 1$ dès lors que p est premier et la réciproque est vraie avec une bonne probabilité. L'exercice précédent fournit un test de primalité similaire au test de Fermat. D'après le Théorème 4.6, son coût est de $\mathrm{O}(\mathsf{M}(d)\log p)$ opérations arithmétiques dans $\mathbb{Z}/p\mathbb{Z}$, soit $\mathrm{O}(\mathsf{M}(d)\mathsf{M}_{\mathbb{Z}}(\log p)\log p)$ opérations binaires.

Exercice 4.7 Soient a et N deux entiers premiers entre eux. Montrer que N est premier *si et seulement si* $\mathrm{X}^{\mathrm{N}} + a = (\mathrm{X}+a)^{\mathrm{N}} \bmod \mathrm{N}$ dans $\mathbb{Z}[\mathrm{X}]$. ■

Exercice 4.8 Montrer que si N est premier, $0 \le a < \mathrm{N}$ et $\mathrm{P}(\mathrm{X}) \in \mathbb{Z}[\mathrm{X}]$, alors $\mathrm{X}^{\mathrm{N}} + a =$

$(X + a)^N \bmod P(X)$ dans $\mathbb{Z}/N\mathbb{Z}[X]$; si de plus, $P(X)$ est de degré $r = O(\log^c N)$, pour un $c > 0$, alors cette égalité peut être testée « en temps polynomial », c'est-à-dire en un nombre d'opérations binaires polynomial en $\log N$. ∎

Notes

La suite de Fibonacci, introduite en 1202 par Leonardo Pisano (mieux connu sous le pseudonyme de Fibonacci) dans un problème récréatif décrivant la croissance d'une population de lapins jouit de riches propriétés algébriques, arithmétiques et combinatoires. Par exemple : (a) F_n est le nombre de façons différentes de paver un rectangle $2 \times (n-1)$ au moyen de dominos 2×1 ; (b) $(F_n)_n$ est une *suite de divisibilité*, c'est-à-dire que F_n divise F_m dès lors que n divise m ; (c) si n est impair, alors

$$F_n = 2^{n-1} \prod_{k=1}^{\frac{n-1}{2}} \left(\frac{1}{4} + \cos^2 \frac{k\pi}{n} \right).$$

Exercice 4.9 Prouver les assertions (a)–(c). ∎

Malgré sa simplicité, la suite de Fibonacci fait l'objet de nombreux problèmes ouverts. Par exemple, on ignore s'il existe une infinité de nombres de Fibonacci premiers. Les nombres de Fibonacci sont omniprésents en mathématiques et en informatique [2] : ils interviennent aussi bien dans l'analyse de l'algorithme d'Euclide pour le calcul du plus grand commun diviseur de deux entiers, que dans la solution négative du dixième problème de Hilbert par Matiyasevich [3].

Historiquement, le premier algorithme rapide pour la division des polynômes est dû à Moenck et Borodin [MB72]. Son point clé est que *le quotient de la division euclidienne de deux polynômes ne dépend que de leurs coefficients de poids fort*. Cette remarque est également à la base du calcul rapide de pgcd, étudié au Chapitre 6.

La paternité de l'algorithme de la Section 4.2 revient à Strassen [Str73]. Une alternative, de même complexité asymptotique, à l'algorithme esquissé en Section 4.2 pour les calculs modulaires a été proposée par Montgomery [Mon85].

Calculer les N premiers termes d'une srlcc de polynôme caractéristique fixé est une opération linéaire en les conditions initiales ; c'est le *dual* de l'opération de division d'un polynôme de degré N par un polynôme fixé. Les conséquences algorithmiques de ce fait seront décrites au Chapitre 12.

Le théorème de Skolem–Mahler affirme que pour toute srlcc (a_n), l'ensemble de ses zéros (les indices i pour lesquels $a_i = 0$) est la réunion d'un ensemble fini et d'un nombre fini de suites arithmétiques. Son étude est une question subtile. Par exemple, déterminer si l'ensemble des zéros est vide est un problème NP-difficile [BP02]. Le livre d'Everest, van der Poorten, Shparlinski et Ward [Eve+03] est une excellente référence pour en savoir plus sur les questions reliées aux srlcc. Sur ces questions, il existe aussi quelques articles synthétiques et bien écrits [CMP87 ; Poo89].

2. Le journal *The Fibonacci Quarterly* est entièrement dédié à l'étude de leurs propriétés.

3. Ce problème proposait de trouver un algorithme pour décider si un système d'équations diophantiennes (polynômes à coefficients entiers) admet une solution en nombres entiers.

L'Exercice 4.1 est le point clé de la méthode de Wiedemann pour la résolution de systèmes linéaires creux, traitée au Chapitre 9.

Le calcul rapide d'un terme de la suite de Fibonacci par exponentiation binaire de la matrice compagnon associée relève du folklore mathématique. Sa généralisation (Exercice 4.4) est décrite par Miller et Brown [MB66], mais était probablement connue bien avant.

Le Théorème 4.9 et le Corollaire 4.11 sont dus à Fiduccia [Fid85] et Shoup [Sho91a]. Leur récente généralisation au cas des matrices polynomiales est à la base du meilleur algorithme pour la résolution de systèmes linéaires à coefficients polynomiaux, exposé au Chapitre 11.

Il n'existe pas de test déterministe de primalité basés sur le test modulaire d'un terme d'une récurrence à coefficients constants. Par contre, on peut caractériser un nombre premier N à l'aide de suites qui vérifient des récurrences à coefficients polynomiaux (comme la factorielle, via le test de Wilson $(N-1)! = -1 \bmod N$). Malheureusement, cela ne fournit pas d'algorithme efficace. Le premier algorithme déterministe qui prouve la primalité en temps polynomial est très récent [AKS04] ; il part de la caractérisation de type Fermat donnée dans l'Exercice 4.8, et exhibe une constante c et un polynôme P tels que la primalité de N est impliquée par la vérification de l'identité de l'Exercice 4.8 pour seulement $r^{1/2} \log N$ valeurs de a.

Bibliographie

AKS04 Agrawal, Manindra, Neeraj Kayal et Nitin Saxena (2004). « PRIMES is in P ». In : *Annals of Mathematics. Second Series*, vol. 160, n°2, p. 781–793.

BP02 Blondel, Vincent D. et Natacha Portier (2002). « The presence of a zero in an integer linear recurrent sequence is NP-hard to decide ». In : *Linear Algebra and its Applications*, vol. 351/352. Fourth special issue on linear systems and control, p. 91–98.

CMP87 Cerlienco, L., M. Mignotte et F. Piras (1987). « Suites récurrentes linéaires. Propriétés algébriques et arithmétiques ». In : *L'Enseignement Mathématique*. II, vol. 33, p. 67–108.

Eve+03 Everest, Graham, Alf van der Poorten, Igor Shparlinski et Thomas Ward (2003). *Recurrence sequences*. Vol. 104. Mathematical Surveys and Monographs. American Mathematical Society.

Fid85 Fiduccia, C. M. (1985). « An efficient formula for linear recurrences ». In : *SIAM Journal on Computing*, vol. 14, n°1, p. 106–112.

MB66 Miller, J. C. P. et D. J. Spencer Brown (1966). « An algorithm for evaluation of remote terms in a linear recurrence sequence ». In : *Computer Journal*, vol. 9, p. 188–190.

MB72 Moenck, R. T. et A. Borodin (1972). « Fast modular transforms via division ». In : *Thirteenth Annual IEEE Symposium on Switching and Automata Theory*, p. 90–96.

Mon85 Montgomery, Peter L. (1985). « Modular multiplication without trial division ». In : *Mathematics of Computation*, vol. 44, n°170, p. 519–521.

Poo89 Poorten, A. J. van der (1989). « Some facts that should be better known, especially about rational functions ». In : *Number theory and applications*. Proceedings of a Conference held at Banff, AB, 1988. Dordrecht : Kluwer, p. 497–528.

Sho91a Shoup, V. (1991). « A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic ». In : *ISSAC'91 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 14–21.

Str73 Strassen, V. (1972/73). « Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten ». In : *Numerische Mathematik*, vol. 20, p. 238–251.

5. Calculs modulaires, évaluation et interpolation

Résumé

Le concept d'algorithme modulaire est central en calcul formel. Il permet de pallier le phénomène d'explosion des expressions intermédiaires. Des cas particuliers importants sont l'évaluation multipoint et l'interpolation, pour lesquels il existe des algorithmes rapides qui peuvent être vus comme des généralisations de la FFT.

5.1 Introduction

Un ingrédient essentiel en calcul formel est l'utilisation de différents types de représentation pour les objets manipulés. Par exemple, un polynôme est classiquement codé par la liste de ses coefficients, mais on peut très bien le représenter par les valeurs qu'il prend en un nombre suffisant de points. D'ailleurs, nous avons vu au Chapitre 2 que c'est bien cette seconde représentation qui est la plus adaptée pour le calcul efficace, via la FFT, du produit de polynômes. Par ailleurs, d'autres opérations (comme la division polynomiale, traitée au Chapitre 4) se font mieux dans la première représentation. D'autres exemples de codages alternatifs déjà rencontrés dans cet ouvrage sont l'écriture des entiers en différentes bases de numération, ou encore la représentation des suites récurrentes linéaires à coefficients constants, soit par leurs éléments, soit par leurs séries génératrices, soit par une récurrence et des conditions initiales.

L'exemple de la FFT montre à quel point il est crucial d'examiner dans quelle représentation un problème donné est plus facile à traiter, et aussi, de trouver des algorithmes rapides pour la conversion d'une représentation à l'autre. Une incarnation de ce concept général est la notion d'*algorithme modulaire*, dont la méthodologie

« évaluation-interpolation » est un cas particulier très important. L'approche modulaire consiste à choisir des *modules* m_i, à faire des calculs modulo chaque m_i et à reconstruire tout à la fin le résultat à l'aide d'une version effective du *théorème des restes chinois*. Pour garantir l'unicité du résultat et la correction de ce schéma, il suffit que les modules soient suffisamment *nombreux et indépendants*. Typiquement, s'il s'agit d'entiers, ils peuvent être choisis premiers entre eux et tels que leur produit dépasse le résultat final. En particulier, pour mettre en place une approche modulaire, on a besoin de bornes *a priori* sur la taille de l'objet à calculer.

Cette approche porte ses fruits surtout lorsque les coefficients dans les calculs intermédiaires sont beaucoup plus gros que ceux du résultat final. Il s'agit du phénomène d'*explosion des expressions intermédiaires*, qui se présente par exemple dans le calcul du déterminant d'une matrice entière ou polynomiale, ou encore au cours du calcul du pgcd de polynômes à coefficients entiers.

Exemple 5.1 — Calcul du déterminant d'une matrice polynomiale. Soit à calculer le déterminant d'une matrice $n \times n$, dont les éléments sont des polynômes de degré au plus d (le cas particulier $d = 1$ correspond au calcul d'un polynôme caractéristique). Le point de départ est la remarque que le pivot de Gauss produit sur la i-ième ligne des fractions rationnelles de degré $2^i d$. Ainsi, sa complexité ne semble pas polynomiale par rapport à n. Une approche évaluation-interpolation résout cette anomalie. Le déterminant étant un polynôme de degré au plus nd, il suffit de choisir un ensemble de $nd + 1$ points, d'y évaluer les éléments de la matrice, de calculer les $nd + 1$ déterminants de matrices scalaires et de finir par une interpolation fournissant le determinant cherché. Le même raisonnement s'applique aux matrices entières.

Choix des modules. Dans certaines situations, le programmeur a le choix des modules. Dans l'exemple précédent, on peut choisir comme modules des polynômes de la forme $X - \omega^i$, dans le cas favorable où l'anneau de base contient une racine principale de l'unité ω d'ordre suffisamment élevé ($\approx 2dn$). En théorie, ce choix est donc possible dès lors que l'anneau de base permet une multiplication polynomiale à base de FFT. En pratique, il existe des plages de degrés pour lesquelles, même si elle est faisable, la FFT n'est pas encore rentable ; ce choix est alors déconseillé. Dans d'autres situations, le choix des modules est intrinsèquement imposé par le problème à traiter ; c'est le cas pour le calcul de la factorielle exposé au Chapitre 15.

5.2 Présentation, résultats

Les problèmes d'évaluation et d'interpolation sont inverses l'un de l'autre. Dans leur version standard, ils s'énoncent ainsi. Soient $a_0, \ldots, a_{n-1}$ des points dans $\mathbb{A}$, où $\mathbb{A}$ est un anneau commutatif.

Évaluation multipoint Étant donné un polynôme P dans $\mathbb{A}[X]$, de degré strictement inférieur à n, calculer les valeurs :

$$P(a_0), \ldots, P(a_{n-1}).$$

Interpolation Étant donnés $b_0, \ldots, b_{n-1} \in \mathbb{A}$, trouver un polynôme $P \in \mathbb{A}[X]$ de degré

strictement inférieur à n tel que

$$\mathrm{P}(a_0) = b_0, \ldots, \mathrm{P}(a_{n-1}) = b_{n-1}.$$

Le problème d'interpolation admet toujours une solution unique sous l'hypothèse technique suivante :

$$a_i - a_j \text{ est inversible dans } \mathbb{A} \text{ lorsque } i \neq j. \tag{H}$$

En effet, si $\mathbf{V} = \left(a_{i-1}^{j-1}\right)_{i,j=1}^{n}$ est la matrice de Vandermonde associée aux points a_i, dont le déterminant vaut $\det(\mathbf{V}) = \prod_{i<j}\left(a_j - a_i\right)$, alors le problème d'interpolation se traduit par la résolution en l'inconnue $\mathbf{p}$ du système linéaire $\mathbf{Vp} = \mathbf{b}$, où $\mathbf{b}$ est le vecteur des b_i. Or, ce système admet une solution unique, puisque l'hypothèse (H) entraîne l'inversibilité du déterminant $\det(\mathbf{V})$ et donc aussi celle de la matrice $\mathbf{V}$.

Exercice 5.1 Montrer que $\det(\mathbf{V}) = \prod_{i<j}(a_j - a_i)$. ∎

Cette discussion suggère un algorithme naïf pour l'interpolation, produisant l'unique solution $\mathbf{p} = \mathbf{V}^{-1}\mathbf{b}$ du système $\mathbf{Vp} = \mathbf{b}$ à l'aide du pivot de Gauss en $\mathrm{O}(n^3)$ opérations dans $\mathbb{A}$, ou encore en $\mathrm{O}(\mathsf{MM}(n)) = \mathrm{O}(n^\theta)$ opérations ($2 \leq \theta < 3$), en utilisant l'algorithmique matricielle rapide (Chapitre 8). De manière analogue, l'évaluation multipoint de P en les a_i se traduit par le produit matrice-vecteur $\mathbf{Vp}$, où $\mathbf{p}$ est le vecteur des coefficients de P ; elle peut donc être effectuée de manière naïve en $\mathrm{O}(n^2)$ opérations arithmétiques. En s'y prenant *vraiment naïvement*, l'interpolation est donc plus coûteuse que l'évaluation multipoint. Nous verrons un peu plus loin qu'une méthode exploitant la *formule d'interpolation de Lagrange* permet de résoudre le problème d'interpolation en complexité *quadratique* en n.

L'objectif de ce chapitre est de montrer qu'il est possible d'atteindre une complexité quasi-optimale, tant pour l'évaluation multipoint que pour l'interpolation, en utilisant des algorithmes de type « diviser pour régner » qui ramènent ces questions à des multiplications de polynômes. Les principaux résultats sont les suivants :

Théorème 5.1 — « Évaluation-interpolation ». On peut effectuer l'évaluation et l'interpolation sur n points vérifiant l'hypothèse (H) en utilisant $\mathrm{O}(\mathsf{M}(n)\log n)$ opérations $(+, -, \times)$ de $\mathbb{A}$. Cette complexité peut être abaissée à $\mathrm{O}(\mathsf{M}(n))$ si les points forment une progression géométrique de raison inversible dans $\mathbb{A}$.

En termes pratiques, si l'anneau de base $\mathbb{A}$ est un corps fini, les algorithmes rapides deviennent avantageux pour des degrés de l'ordre de quelques dizaines : c'est sensiblement mieux que pour les algorithmes rapides de type « diviser pour régner » utilisés aux Chapitres 6 et 7 pour le calcul rapide de pgcd ou d'approximants de Padé–Hermite ; cela reflète une relative simplicité des algorithmes.

L'évaluation multipoint et l'interpolation sont des instances particulières des problèmes *modules multiples* et *restes chinois* s'énonçant comme suit :

Modules multiples Étant donnés des polynômes $\mathrm{P}, m_1, \ldots, m_r$ dans $\mathbb{A}[\mathrm{X}]$ qui satisfont $\deg(\mathrm{P}) < n = \sum_i \deg(m_i)$, calculer les restes :

$$\mathrm{P} \bmod m_1, \ldots, \mathrm{P} \bmod m_r.$$

Restes chinois Étant donnés des polynômes $b_1, \ldots, b_r, m_1, \ldots, m_r$ dans $\mathbb{A}[X]$, avec m_i unitaires, trouver $P \in \mathbb{A}[X]$ de degré strictement inférieur à $n = \sum_i \deg(m_i)$ tel que

$$P \bmod m_1 = b_1, \ldots, P \bmod m_r = b_r.$$

Les techniques de ce chapitre s'y généralisent. Pour les restes chinois, l'unicité du résultat est garantie par l'hypothèse que les résultants $\mathrm{Res}(m_i, m_j)$ sont inversibles dans $\mathbb{A}$, pour tous $i \neq j$. Cette condition technique généralise la condition d'inversibilité des $a_i - a_j$ (se rapporter au Chapitre 6 pour la définition du résultant). On peut alors prouver le résultat suivant, laissé en exercice :

Théorème 5.2 On peut résoudre les deux problèmes ci-dessus avec un algorithme qui utilise $O(M(n)\log n)$ opérations $(+, -, \times)$ dans $\mathbb{A}$.

Naturellement, on peut poser les questions précédentes dans le cas où l'anneau de polynômes $\mathbb{A}[X]$ est remplacé par l'anneau $\mathbb{Z}$, les polynômes m_i sont remplacés par des modules entiers a_i. Il est possible d'obtenir le même type de résultats de complexité, cette fois en comptant les opérations binaires.

Théorème 5.3 Soient $a_1, \ldots, a_r$ des entiers strictement positifs de produit $A = a_1 \cdots a_r$ ayant au plus n chiffres. Si b est un entier d'au plus n chiffres, alors on peut calculer $b \bmod a_1, \ldots, b \bmod a_r$ en $O(M_{\mathbb{Z}}(n)\log n)$ opérations binaires.

Inversement, sous l'hypothèse que les a_i sont premiers entre eux, à partir d'entiers $b_1, \ldots, b_r$ tels que $b_i < a_i$, on peut calculer l'unique entier $b < A$ tel que $b \bmod a_i = b_i$ pour tout i, en $O(M_{\mathbb{Z}}(n)\log n)$ opérations binaires.

La suite de ce chapitre est organisée comme suit : dans la Section 5.3 est décrite la méthode d'interpolation de Lagrange, de complexité quadratique. La Section 5.4 est consacrée aux algorithmes rapides sous-jacents à la preuve du Théorème 5.1. Comme les techniques pour le cas des entiers sont les mêmes que pour les polynômes, nous laisserons la preuve du Théorème 5.3 à titre d'exercice.

5.3 Interpolation de Lagrange

Un algorithme de complexité quadratique pour l'interpolation polynomiale repose sur une écriture explicite du polynôme interpolant. Soient b_i les valeurs à interpoler et a_i les points d'interpolation vérifiant l'hypothèse (H). On peut alors écrire P sous la forme :

$$P(X) = \sum_{i=0}^{n-1} b_i \prod_{0 \leq j \leq n-1,\ j \neq i} \frac{X - a_j}{a_i - a_j}.$$

Cette égalité est usuellement appelée la *formule d'interpolation de Lagrange*. Pour la prouver, il suffit d'observer que pour tout i, le produit s'annule en a_j $(j \neq i)$, et vaut 1 en a_i. Afin de simplifier l'écriture de la formule de Lagrange, posons

$$A = \prod_j (X - a_j) \quad \text{et} \quad A_i = \prod_{j \neq i} (X - a_j) = \frac{A}{X - a_i},$$

Entrée $a_0,\dots,a_{n-1} \in \mathbb{A}$ vérifiant (H) et $b_0,\dots,b_{n-1}$ dans $\mathbb{A}$.
Sortie L'unique polynôme $P \in \mathbb{A}[X]$ de degré strictement inférieur à n tel que $P(a_i) = b_i$ pour tout i.

1. Initialiser A à 1, et P à 0.
2. Pour $i = 0,\dots,n-1$ faire
   $A = A \cdot (X - a_i)$.
3. Pour $i = 0,\dots,n-1$ faire
   $A_i = A/(X - a_i)$
   $q_i = A_i(a_i)$
   $P = P + b_i A_i/q_i$.
4. Renvoyer P.

Algorithme 5.1 – Algorithme de Lagrange pour l'interpolation polynomiale.

pour $i = 0,\dots,n-1$. On a donc les relations $A(a_i) = 0$ et pour $j \neq i$, $A_i(a_j) = 0$. On obtient alors l'écriture

$$P = \sum_{i=0}^{n-1} b_i \frac{A_i(X)}{A_i(a_i)}.$$

Une approche directe pour l'interpolation revient à calculer tout d'abord les polynômes A_i, puis leurs valeurs en les points a_i, et enfin à effectuer les combinaisons linéaires nécessaires. Le seul point non trivial est le calcul des A_i : si on n'y fait pas attention, on risque de sortir des bornes en $O(n^2)$ (par exemple, si on les calcule tous indépendamment, et chacun de manière quadratique). Pour faire mieux, on partage le gros des calculs, en calculant d'abord le polynôme A.

Proposition 5.4 L'Algorithme 5.1 utilise $O(n^2)$ opérations dans $\mathbb{A}$.

Démonstration. La multiplication d'un polynôme de degré d par un polynôme de degré 1 prend un temps linéaire en d, disons Cd opérations, C étant une constante. Calculer A demande donc $C(1 + 2 + \cdots + (n-1)) = O(n^2)$ opérations. Ensuite, chaque passage dans la seconde boucle prend un temps linéaire en n. Il y a n passages, d'où à nouveau un coût quadratique. ■

5.4 Algorithmes rapides

Les idées

L'idée fondamentale de l'algorithme d'évaluation multipoint rapide est d'utiliser une technique de type « diviser pour régner ». Supposons pour simplifier que n soit pair et séparons l'ensemble des points $a_0,\dots,a_{n-1}$ en deux paquets, $a_0,\dots,a_{n/2-1}$ et $a_{n/2},\dots,a_{n-1}$. Il est naturel d'associer à P deux polynômes P_0 et P_1 de degrés strictement inférieurs à $n/2$, tels que :

$$P_0(a_0) = P(a_0),\dots,P_0(a_{n/2-1}) = P(a_{n/2-1}) \quad \text{et} \quad P_1(a_{n/2}) = P(a_{n/2}),\dots,P_1(a_{n-1}) = P(a_{n-1}).$$

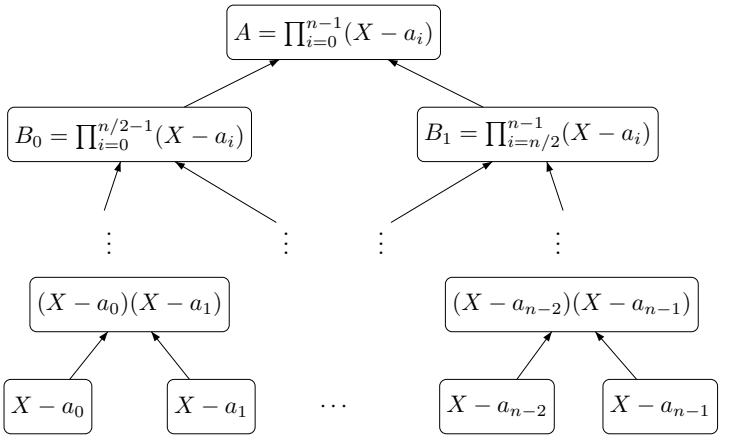


Figure 5.2 – Arbre des sous-produits.

Entrée $a_0, \ldots, a_{n-1}$ dans $\mathbb{A}$, avec n de la forme $n = 2^\kappa$ ($\kappa \in \mathbb{N}$).
Sortie L'arbre $\mathcal{B}$ des sous-produits associé aux points a_i.

1. Si $n = 1$, renvoyer $X - a_0$.
2. Calculer récursivement l'arbre $\mathcal{B}_0$ des sous-produits associé à $a_0, \ldots, a_{n/2-1}$.
3. Calculer récursivement l'arbre $\mathcal{B}_1$ des sous-produits associé à $a_{n/2}, \ldots, a_{n-1}$.
4. Calculer $A = B_0 B_1$, où B_0 et B_1 sont respectivement les racines de $\mathcal{B}_0$ et de $\mathcal{B}_1$.
5. Renvoyer l'arbre de racine A et dont les fils sont $\mathcal{B}_0$ et $\mathcal{B}_1$.

Algorithme 5.3 – Algorithme rapide pour le calcul de l'arbre des sous-produits.

Pour effectuer cette construction, le choix suivant s'impose :

$$P_0 = P \bmod (X - a_0) \cdots (X - a_{n/2-1}), \quad P_1 = P \bmod (X - a_{n/2}) \cdots (X - a_{n-1}).$$

On a donc ramené le calcul en degré n à deux calculs en degré $n/2$, plus deux divisions euclidiennes ; on va ensuite itérer ce processus. En bout de course, on retrouve les valeurs $P(a_i)$, car celles-ci peuvent également s'écrire $P \bmod (X - a_i)$.

Remarquons que les polynômes par lesquels on effectue les divisions euclidiennes ne sont pas « gratuits » : on doit les (pré)calculer. Pour cela, l'idée est de les empiler dans une structure d'arbre binaire, dont les nœuds internes sont étiquetés par des polynômes.

L'arbre des sous-produits

Pour simplifier, dans tout ce qui suit, on suppose que le nombre de points est une puissance de 2, $n = 2^\kappa$. Quand ce n'est pas le cas, on adapte toutes les constructions (on fabrique toujours un arbre binaire, mais il lui manque alors des nœuds internes

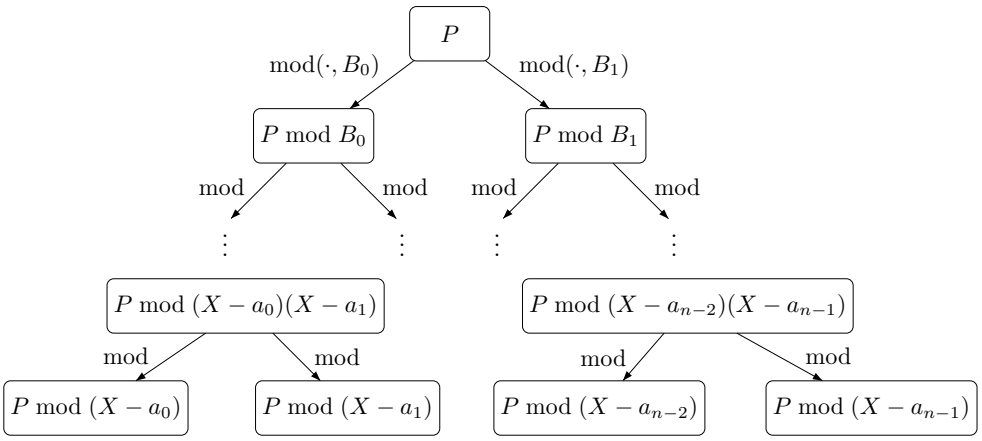


FIGURE 5.4 – Parcours descendant de l'arbre des sous-produits.

à certains niveaux). On peut donner la définition récursive suivante pour cet arbre (noté $\mathcal{B}$), représenté dans la Figure 5.2 :

— Si $\kappa = 0$, $\mathcal{B}$ se réduit à un nœud unique qui contient le polynôme $X - a_0$.
— Sinon, soient $\mathcal{B}_0$ et $\mathcal{B}_1$ les arbres associés respectivement aux points $a_0, \ldots, a_{2^{\kappa-1}-1}$ et $a_{2^{\kappa-1}}, \ldots, a_{2^\kappa-1}$. Soient B_0 et B_1 les polynômes présents aux racines de $\mathcal{B}_0$ et $\mathcal{B}_1$. Alors $\mathcal{B}$ est l'arbre dont la racine contient le produit $B_0 B_1$ et dont les fils sont $\mathcal{B}_0$ et $\mathcal{B}_1$.

L'intérêt de cette structure d'arbre est double : d'une part, il contient tous les polynômes par lesquels on va diviser lors de l'algorithme d'évaluation, d'autre part, le coût de son calcul s'amortit, et devient essentiellement linéaire en le degré.

Proposition 5.5 L'Algorithme 5.3 calcule tous les polynômes apparaissant dans l'arbre $\mathcal{B}$ en $O(M(n)\log n)$ opérations arithmétiques dans $\mathbb{A}$.

Démonstration. Soit $T(n)$ le coût du calcul de l'arbre pour n points. La définition récursive implique l'inégalité suivante pour $T(n)$:

$$T(n) \leq 2T\left(\frac{n}{2}\right) + M\left(\frac{n}{2}\right).$$

Le résultat s'ensuit en invoquant le théorème « diviser pour régner ». ■

Évaluation multipoint rapide

L'algorithme d'évaluation multipoint devient simple à écrire de manière récursive, si on s'autorise un peu de souplesse dans l'écriture, en supposant précalculés tous les nœuds de l'arbre. Comme illustré dans la Figure 5.4, cela correspond à faire *descendre* les restes de divisions euclidiennes dans l'arbre des sous-produits. Or, la division avec reste par un polynôme unitaire de degré n peut s'effectuer en $O(M(n))$ opérations de $\mathbb{A}$ (Chapitre 4). Nous obtenons le résultat de complexité suivant :

Entrée $a_0,\dots,a_{n-1}$ dans $\mathbb{A}$, avec n de la forme $n = 2^\kappa$ ($\kappa \in \mathbb{N}$), l'arbre $\mathcal{B}$ des sous-produits associé aux points a_i, P dans $\mathbb{A}[X]$, avec $\deg P < n$.
Sortie $P(a_0),\dots,P(a_{n-1})$.

1. Si $n = 1$, renvoyer P.
2. $P_0 = P \bmod (X-a_0)\cdots(X-a_{n/2-1})$.
3. $P_1 = P \bmod (X-a_{n/2})\cdots(X-a_{n-1})$.
4. Calculer (par un appel récursif) $L_0 = P_0(a_0),\dots,P_0(a_{n/2-1})$.
5. Calculer (par un appel récursif) $L_1 = P_1(a_{n/2}),\dots,P_1(a_{n-1})$.
6. Renvoyer L_0, L_1.

Algorithme 5.5 – Algorithme rapide d'évaluation multipoint par division répétée.

Proposition 5.6 La complexité de l'Algorithme 5.5 est de $O(M(n)\log n)$ opérations arithmétiques dans $\mathbb{A}$.

Démonstration. Soit $T(n)$ le coût du calcul pour n points. L'algorithme ci-dessus implique la récurrence suivante pour $T(n)$:

$$T(n) \le 2T\left(\frac{n}{2}\right) + O(M(n)),$$

et le résultat en découle grâce au théorème « diviser pour régner ». ∎

Exercice 5.2 Si $\omega \in \mathbb{A}$ est une racine principale n-ième de l'unité, montrer que l'algorithme FFT vu au Chapitre 2 est un cas particulier de l'Algorithme 5.5. ∎

Exercice 5.3 Estimer la complexité de la variante de l'Algorithme 5.5 dans laquelle l'arbre des sous-produits n'est pas précalculé, les polynômes modulo lesquels s'effectuent les divisions étant calculés à la volée. ∎

Sommes de fractions

Un problème intermédiaire à traiter avant de résoudre efficacement le problème d'interpolation est celui du calcul rapide d'une somme de fractions. Rappelons les définitions introduites dans la Section 5.3. À partir des points a_i, nous y avions défini les polynômes

$$A = \prod_j (X-a_j) \qquad \text{et} \qquad A_i = \prod_{j\neq i} (X-a_j) = \frac{A}{X-a_i}.$$

Il était apparu que pour effectuer l'interpolation, il suffisait de savoir effectuer la tâche suivante : étant données des valeurs c_i, calculer le numérateur et le dénominateur de la fraction rationnelle

$$\sum_{i=0}^{n-1} \frac{c_i}{X-a_i}. \tag{5.1}$$

Entrée $a_0,\dots,a_{n-1} \in \mathbb{A}$ et $c_0,\dots,c_{n-1}$ dans $\mathbb{A}$, avec n de la forme $n = 2^\kappa$ ($\kappa \in \mathbb{N}$).
Sortie $N, P \in \mathbb{A}[X]$ tels que $P = (X-a_0)\cdots(X-a_{n-1})$ et $\frac{N}{P} = \sum_{i=0}^{n-1} \frac{c_i}{X-a_i}$.

1. Si $n = 1$, alors renvoyer c_0 et $X - a_0$.
2. Appeler récursivement l'algorithme avec $a_0,\dots,a_{n/2-1}$ et $c_0,\dots,c_{n/2-1}$. On note N_1 et P_1 les polynômes renvoyés.
3. Appeler récursivement l'algorithme avec $a_{n/2},\dots,a_{n-1}$ et $c_{n/2},\dots,c_{n-1}$ pour obtenir en sortie N_2 et P_2.
4. Renvoyer $N_1P_2 + N_2P_1$ et P_1P_2.

Algorithme 5.6 – Somme de fractions.

C'est à cette question que nous allons répondre maintenant ; à nouveau, on utilise une idée de type « diviser pour régner » : par deux appels récursifs, on calcule

$$S_1 = \sum_{i=0}^{n/2-1} \frac{c_i}{X-a_i} \quad \text{et} \quad S_2 = \sum_{i=n/2}^{n} \frac{c_i}{X-a_i}$$

et on renvoie $S = S_1 + S_2$. Soit $T(n)$ le coût du calcul pour n points. La méthode est résumée dans l'Algorithme 5.6. Son coût satisfait à la récurrence :

$$T(n) \le 2T\left(\frac{n}{2}\right) + O(M(n)),$$

d'où l'on déduit le résultat suivant, en faisant encore une fois appel au théorème « diviser pour régner ».

Proposition 5.7 L'Algorithme 5.6 calcule le dénominateur et le numérateur de la fraction rationnelle (5.1) en $O(M(n)\log n)$ opérations dans $\mathbb{A}$.

Interpolation

À ce stade, l'interpolation ne pose plus de réelle difficulté. Au vu des formules de la Section 5.3, le polynôme interpolant les valeurs b_i aux points a_i est donné par :

$$P(X) = \sum_{i=0}^{n-1} b_i \frac{A_i(X)}{A_i(a_i)} = A(X) \times \sum_{i=0}^{n-1} \frac{b_i/A_i(a_i)}{X-a_i}.$$

D'après les paragraphes précédents, on sait comment calculer rapidement la fraction rationnelle $\sum c_i/(X-a_i)$; il ne reste plus qu'à calculer les constantes $c_i = b_i/A_i(a_i)$. Le calcul itératif des constantes c_i reviendrait à évaluer n polynômes différents, chacun en un point, et aurait donc une complexité trop élevée, quadratique en n. La proposition suivante permet de remplacer ces n évaluations par une évaluation multipoint d'un unique polynôme en n points, ce qui mène à un gain de complexité.

Entrée $a_0,\dots,a_{n-1} \in \mathbb{A}$ vérifiant (H) et $b_0,\dots,b_{n-1}$ dans $\mathbb{A}$.
Sortie L'unique polynôme $P \in \mathbb{A}[X]$ de degré strictement inférieur à n tel que $P(a_i) = b_i$ pour tout i.

1. Calculer l'arbre des sous-produits associé aux points a_i (de racine A).
2. Calculer $d_i = A'(a_i)$ pour tout $0 \le i \le n-1$, avec l'Algorithme 5.5.
3. $(c_0,\dots,c_{n-1}) = (b_0/d_0,\dots,b_{n-1}/d_{n-1})$.
4. Calculer la somme R de fractions
$$c_0/(X-a_0)+\cdots+c_{n-1}/(X-a_{n-1})$$
avec l'Algorithme 5.6.
5. Renvoyer le numérateur de R.

Algorithme 5.7 – Algorithme d'interpolation rapide.

Proposition 5.8 Pour tout $i = 0,\dots,n-1$, on a $A_i(a_i) = A'(a_i)$.

Démonstration. On dérive A, ce qui fournit l'égalité :
$$A' = \sum_{i=0}^{n-1} \prod_{j\neq i} (X-a_j) = \sum_{i=0}^{n-1} A_i.$$
On a vu précédemment que $A_i(a_j) = 0$ pour $j \neq i$, ce qui donne le résultat. ■

L'algorithme d'interpolation et l'estimation de son coût s'en déduisent facilement.

Proposition 5.9 La complexité de l'Algorithme 5.7 est $O(M(n)\log n)$ opérations de $\mathbb{A}$.

Démonstration. C'est une conséquence des Propositions 5.5, 5.6 et 5.7. ■

Évaluation et interpolation sur une suite géométrique

Dans cette section nous montrons que tout polynôme de degré n peut être évalué et interpolé sur des points en progression géométrique $\{1,q,\dots,q^{n-1}\}$ en $O(M(n))$ opérations.

Proposition 5.10 Soit q un élément inversible de $\mathbb{A}$, et $a_i = q^i$ pour $0 \le i < n$. Alors :

1. On peut évaluer tout polynôme $P \in \mathbb{A}[X]$ de degré strictement inférieur à n sur les points $a_0,\dots,a_{n-1}$ en $O(M(n))$ opérations dans $\mathbb{A}$.
2. Si les points (a_i) vérifient l'hypothèse (H) on peut effectuer l'interpolation en les points $a_0,\dots,a_{n-1}$ en $O(M(n))$ opérations dans $\mathbb{A}$.

Démonstration. Soit $P = p_0 + p_1 X + \cdots + p_{n-1} X^{n-1}$. Pour $i = 0, \ldots, 2n-2$, on introduit les nombres triangulaires $t_i = i(i-1)/2$ et la suite $\beta_i = q^{t_i}$; pour $i = 0, \ldots, n-1$ on construit $\gamma_i = p_i/\beta_i$. Tous les éléments q^{t_i}, γ_i et β_i peuvent être calculés en $O(n)$ opérations, en utilisant l'égalité $q^{t_{i+1}} = q^i q^{t_i}$. L'observation clé est que $ij = t_{i+j} - t_i - t_j$ et donc $q^{ij} = \beta_i^{-1}\beta_j^{-1}\beta_{i+j}$. L'algorithme d'évaluation repose alors sur la formule

$$P(q^i) = \sum_{j=0}^{n-1} p_j q^{ij} = \beta_i^{-1} \cdot \sum_{j=0}^{n-1} \gamma_j \beta_{i+j},$$

qui montre que les valeurs $P(q^i)$ sont, à des facteurs constants près, données par les coefficients de $X^{n-1}, \ldots, X^{2n-2}$ dans le produit de $\sum_{i=0}^{n-1} \gamma_i X^{n-i-1}$ et $\sum_{i=0}^{2n-2} \beta_i X^i$.

Pour l'interpolation, on adapte l'Algorithme 5.7, de façon à économiser un facteur log en tirant parti de la structure des points $a_i = q^i$.

On commence par remarquer que *la racine* $A(X)$ de l'arbre des sous-produits peut être calculée en $O(M(n))$ opérations dans $\mathbb{A}$, par un algorithme *qui ne nécessite pas la construction de tout l'arbre des sous-produits*, donc différent de l'Algorithme 5.3. En effet, puisque les points a_i forment une suite géométrique, il est possible de calculer récursivement $A = \prod_{i=0}^{n-1}(X - a_i)$ par *un seul appel récursif* calculant $B_0 = \prod_{i=0}^{n/2-1}(X - a_i)$, suivi du calcul de $B_1 = \prod_{i=n/2}^{n-1}(X - a_i)$ à partir de B_0 par une homothétie de coût $O(n)$, et enfin du calcul du produit $A = B_0 B_1$.

Par ailleurs, observons que dans l'Algorithme 5.7, le calcul de tout l'arbre des sous-produits a été utilisé *uniquement* pour l'évaluation multipoint de A'. Or, par la première partie de la Proposition 5.10, l'évaluation de A' sur les points a_i peut s'effectuer en $O(M(n))$ opérations.

Il nous reste à prouver que la somme des fractions (5.1), qui devient ici

$$\frac{N}{A} = \sum_{i=0}^{n-1} \frac{c_i}{X - q^i} \tag{5.2}$$

peut également être déterminée pour le même coût.

On adopte la stratégie suivante : on calcule le développement en série à l'ordre n de la fraction (5.2). Celui-ci suffit pour en déduire son numérateur $N(X)$. Pour ce faire, on utilise la formule $1/(a - X) = \sum_{j \geq 0} a^{-j-1} X^j$ valable dans $\mathbb{A}[[X]]$ pour tout $a \in \mathbb{A}$ inversible. On obtient :

$$\sum_{i=0}^{n-1} \frac{c_i}{X - q^i} \mod X^n = -\sum_{i=0}^{n-1} \left(\sum_{j=0}^{n-1} c_i q^{-i(j+1)} X^j \right) = -\sum_{j=0}^{n-1} C(q^{-j-1}) X^j,$$

où $C(X)$ est le polynôme $\sum_{i=0}^{n-1} c_i X^i$. Or, les points $q^{-1}, q^{-2}, \ldots, q^{-n}$ étant en progression géométrique, on sait évaluer C sur ces points en $O(M(n))$. ■

Exercice 5.4 Montrer que tout l'arbre des sous-produits associé aux points $a_i = q^i$, $i = 0, \ldots, n-1$, peut être calculé en $M(n) + O(n \log n)$ opérations dans $\mathbb{A}$. ■

Une approche évaluation-interpolation reposant sur la Proposition 5.10 permet d'obtenir un bon algorithme pour la multiplication des matrices polynomiales.

Corollaire 5.11 Soit $\mathbb{K}$ un corps possédant au moins $2d$ éléments. Il est possible de multiplier deux matrices polynomiales de $\mathcal{M}_n(\mathbb{K}[X])$ de degré au plus d en $\mathsf{MM}(n,d) = O(d\,\mathsf{MM}(n) + n^2\,\mathsf{M}(d))$ opérations dans $\mathbb{K}$.

Exercices

Exercice 5.5 — Calcul rapide de sommes de puissances. Soient $x_1, x_2, \ldots, x_n$ des éléments de $\mathbb{K}$ et soit $P_k = \sum_{i=1}^{n} x_i^k$ pour $k \geq 1$. Montrer que, à partir de la donnée des $x_1, \ldots, x_n$, il est possible de calculer :

1. tous les P_k dont les indices $k \leq n$ sont une puissance de 2 en $O(n \log n)$ opérations dans $\mathbb{K}$.
2. tous les $P_1, \ldots, P_n$ en $O(\mathsf{M}(n) \log n)$ opérations dans $\mathbb{K}$.

■

Exercice 5.6 Estimer la complexité de l'algorithme direct pour évaluer un polynôme de degré au plus n et ses premières n dérivées en un point. Imaginer un algorithme de complexité quasi-linéaire résolvant ce problème. ■

Exercice 5.7 Soient $a, b, c, d \in \mathbb{A}$. Montrer qu'on peut évaluer tout polynôme de degré au plus n en $\{ba^{2i} + ca^i + d, \quad 0 \leq i < n\}$, en $O(\mathsf{M}(n))$ opérations dans $\mathbb{A}$. ■

Exercice 5.8 Soit $\mathbb{K}$ un corps, $m_1, m_2 \in \mathbb{K}[X]$ des polynômes premiers entre eux, de degrés au plus n, et $v_1, v_2 \in \mathbb{K}[X]$ de degrés strictement inférieurs à n. Donner un algorithme de complexité $O(\mathsf{M}(n) \log n)$ qui calcule $f \in \mathbb{K}[X]$ de degré strictement inférieur à $2n$ tel que $f = v_1 \bmod m_1$ et $f = v_2 \bmod m_2$. [Indication : utiliser un pgcd étendu.]

En déduire un algorithme de type « diviser pour régner » pour l'interpolation polynomiale en degré n, de complexité $O(\mathsf{M}(n) \log^2 n)$. ■

Exercice 5.9 — Multiplication de polynômes à deux variables par substitution de Kronecker. Soient f et g dans $\mathbb{K}[X, Y]$ de degrés au plus d_X en X et au plus d_Y en Y.

1. Montrer qu'il est possible de calculer tous les coefficients de $h = fg$ avec un algorithme de complexité $O(\mathsf{M}(d_X d_Y))$ opérations dans $\mathbb{K}$. Utiliser la substitution $X \leftarrow Y^{2d_Y+1}$ pour se ramener à une multiplication de polynômes à une variable.
2. Améliorer ce résultat en donnant un schéma d'évaluation-interpolation qui permette le calcul des coefficients de h en $O(d_X \mathsf{M}(d_Y) + d_Y \mathsf{M}(d_X))$ opérations de $\mathbb{K}$ si $\mathbb{K}$ est assez grand.

■

Notes

Pour évaluer un polynôme $P(X) = p_{n-1}X^{n-1} + \cdots + p_0$ en un seul point a, le *schéma de Horner* $P(a) = (\cdots((p_{n-1}a + p_{n-2})a + p_{n-3})a + \cdots + p_0)$ minimise le nombre d'opérations (additions ou multiplications). Cet algorithme effectue $n-1$ additions et

$n-1$ multiplications dans $\mathbb{A}$, et on ne peut pas faire mieux en général (ceci, si les coefficients du polynôme sont algébriquement indépendants — c'est-à-dire s'ils se comportent comme des indéterminées). Ce résultat d'optimalité a été conjecturé par Ostrowski [Ost54] et prouvé par Pan [Pan66]. Par contre, pour un polynôme donné, il peut être possible de faire mieux : l'évaluation de $P(X) = X^{n-1}$ se fait en temps logarithmique en n.

Une forme particulière du théorème des restes chinois a été enoncée il y a plus de 2000 ans par le mathématicien chinois Sun Tsu. Le traitement moderne est dû à Gauss [Gau66]. Ce théorème admet une formulation très générale, en termes d'idéaux d'anneaux non nécessairement commutatifs : Si R est un anneau et $I_1, \ldots, I_r$ des idéaux (à gauche) de R mutuellement premiers (c'est-à-dire $I_i + I_j = R$ pour $i < j$), alors l'anneau quotient $R/\cap_i I$ est isomorphe à l'anneau produit $\prod R/I_i$ via l'isomorphisme $x \bmod I \mapsto (x \bmod I_1, \ldots, x \bmod I_r)$. La formule d'interpolation de Lagrange est due à Waring [War79]. Les avantages pratiques de l'arithmétique modulaire et les premières applications en informatique du théorème des restes chinois ont été mis en évidence dans les années 1950 par Svoboda et Valach [SV55], et indépendamment par Garner [Gar59] et Takahasi et Ishibashi [TI61].

Le premier algorithme sous-quadratique, de complexité arithmétique $O(n^{1,91})$, pour l'évaluation multipoint est dû à Borodin et Munro [BM71] et repose sur une approche *pas de bébés / pas de géants* exploitant la multiplication sous-cubique des matrices de Strassen [Str69]. Horowitz [Hor72] a étendu ce résultat à l'interpolation. La Proposition 5.5 calcule efficacement les fonctions symétriques élémentaires de n éléments $a_0, \ldots, a_{n-1}$ de $\mathbb{A}$; elle est due également à Horowitz [Hor72]. S'inspirant de la FFT, Fiduccia [Fid72b] eut l'idée d'un algorithme pour l'évaluation multipoint à base de division polynomiale récursive. Ne disposant pas de division de complexité sous-quadratique, son algorithme récursif reste quadratique. Borodin et Moenck corrigent ce point dans deux articles successifs [BM74 ; MB72], reposant sur des algorithmes quasi-optimaux pour la division [MB72 ; Str73]. Montgomery [Mon92] améliore la constante dans la complexité $O\big(M(n)\log n\big)$ de l'évaluation multipoint, sous l'hypothèse que la FFT est utilisée pour la multiplication polynomiale. L'algorithme d'évaluation sur une suite géométrique exposé dans ce chapitre provient des travaux de Rabiner, Schafer, Rader [RSR69], et Bluestein [Blu70]. À l'origine, cet algorithme a permis de montrer qu'une DFT_ω, pour ω racine primitive n-ième de l'unité, peut être effectuée en $O(n \log n)$ opérations même si n n'est pas une puissance de 2. L'algorithme d'interpolation est une adaptation de celui de Mersereau [Mer74]. Les meilleures constantes actuellement connues dans les $O(\cdot)$ du Théorème 5.1 sont obtenus à l'aide du principe de transposition de Tellegen [BLS03 ; BS05].

On connaît peu de choses sur la complexité intrinsèque de l'évaluation multipoint et de l'interpolation en degré n. Strassen [Str73] a prouvé une borne inférieure de type $n \log n$. Shoup et Smolensky [SS97] ont montré que ces bornes restent essentiellement vraies même dans un modèle où des précalculs en quantité illimitée sur les points d'évaluation sont permis. Cependant, l'optimalité des meilleurs algorithmes connus, de complexité $O(n \log^2 n)$, reste un problème ouvert. Même dans le cas particulier où les points d'évaluation sont en progression arithmétique, on ne dispose d'aucun algorithme d'évaluation multipoint de complexité $O(M(n))$, ni d'une preuve qu'un tel algorithme n'existe pas. De même, pour des points quelconques $a_0, \ldots, a_{n-1}$ on ne

connaît pas d'algorithme de complexité $O(M(n))$ pour calculer le polynôme $\prod_i (X - a_i)$. Notons toutefois que de tels algorithmes existent dans les cas particuliers où les a_i forment une progression arithmétique ou géométrique [BS05]. L'Exercice 5.8 provient de [HH71] ; historiquement, ce fut le premier algorithme rapide pour les restes chinois. Les Exercices 5.6 et 5.7 sont tirés de [ASU75].

Bibliographie

ASU75 Aho, A. V., K. Steiglitz et J. D. Ullman (1975). « Evaluating polynomials at fixed sets of points ». In : *SIAM Journal on Computing*, vol. 4, n°4, p. 533–539.

BLS03 Bostan, Alin, Grégoire Lecerf et Éric Schost (2003). « Tellegen's principle into practice ». In : *ISSAC'03 : International Symposium on Symbolic and Algebraic Computation*. Éd. par J. R. Sendra. ACM Press, p. 37–44.

Blu70 Bluestein, L. I. (1970). « A linear filtering approach to the computation of the discrete Fourier transform ». In : *IEEE Trans. Electroacoustics*, vol. AU-18, p. 451–455.

BM71 Borodin, A. et I. Munro (1971). « Evaluating polynomials at many points ». In : *Information Processing Letters*, vol. 1, n°2, p. 66–68.

BM74 Borodin, A. et R. T. Moenck (1974). « Fast modular transforms ». In : *Journal of Computer and System Sciences*, vol. 8, n°3, p. 366–386.

BS05 Bostan, Alin et Éric Schost (2005). « Polynomial evaluation and interpolation on special sets of points ». In : *Journal of Complexity*, vol. 21, n°4, p. 420–446.

Fid72b Fiduccia, C. M. (1972). « Polynomial evaluation via the division algorithm : the fast Fourier transform revisited. » In : *STOC'72 : ACM Symposium on Theory of Computing*, p. 88–93.

Gar59 Garner, Harvey L. (1959). « The residue number system ». In : *IRE-AIEE-ACM'59 Western joint computer conference*. San Francisco, California : ACM, p. 146–153.

Gau66 Gauss, Carl Friedrich (1966). *Disquisitiones arithmeticae*. Translated into English by Arthur A. Clarke, S. J. Yale University Press.

HH71 Heindel, L. E. et E. Horowitz (1971). « On decreasing the computing time for modular arithmetic ». In : *12th symposium on switching and automata theory*. IEEE, p. 126–128.

Hor72 Horowitz, E. (1972). « A fast method for interpolation using preconditioning ». In : *Information Processing Letters*, vol. 1, n°4, p. 157–163.

MB72 Moenck, R. T. et A. Borodin (1972). « Fast modular transforms via division ». In : *Thirteenth Annual IEEE Symposium on Switching and Automata Theory*, p. 90–96.

Mer74 Mersereau, Russell M. (1974). « An algorithm for performing an inverse chirp z-transform ». In : *IEEE Transactions on Audio and Electroacoustics*, vol. ASSP-22, n°5, p. 387–388.

Mon92 Montgomery, P. L. (1992). « An FFT extension of the elliptic curve method of factorization ». Thèse de doctorat. University of California, Los Angeles CA.

Ost54 Ostrowski, A. M. (1954). « On two problems in abstract algebra connected with Horner's rule ». In : *Studies in mathematics and mechanics presented to Richard von Mises*. NY : Academic Press Inc., p. 40–48.

Pan66 Pan, V. Ya. (1966). « Methods of computing values of polynomials ». In : *Russian Mathematical Surveys*, vol. 21, n°1, p. 105–136.

RSR69 Rabiner, L. R., R. W. Schafer et C. M. Rader (1969). « The chirp *z*-transform algorithm and its application ». In : *Bell System Technical Journal*, vol. 48, p. 1249–1292.

SS97 Shoup, Victor et Roman Smolensky (1996/97). « Lower bounds for polynomial evaluation and interpolation problems ». In : *Computational Complexity*, vol. 6, n°4, p. 301–311.

Str69 Strassen, V. (1969). « Gaussian elimination is not optimal ». In : *Numerische Mathematik*, vol. 13, p. 354–356.

Str73 — (1972/73). « Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten ». In : *Numerische Mathematik*, vol. 20, p. 238–251.

SV55 Svoboda, A. et M. Valach (1955). « Operátorové obvody (Operational circuits) ». In : *Stroje na Zpracování Informací (Information Processing Machines)*, vol. 3, p. 247–295.

TI61 Takahasi, H. et Y. Ishibashi (1961). « A new method for exact calculation by a digital computer ». In : *Information Processing in Japan*, p. 28–42.

War79 Waring, E. (1779). « Problems concerning interpolations ». In : *Philosophical Transactions of the Royal Society of London*, vol. 59, p. 59–67.

6. Pgcd et résultant

Résumé

Les calculs de pgcd sont cruciaux pour la simplification des fractions, qu'il s'agisse de fractions d'entiers ou de fractions de polynômes. Les algorithmes efficaces de factorisation de polynômes reposent par ailleurs de manière essentielle sur des calculs de pgcd. Comme pour la multiplication, les algorithmes efficaces de pgcd sont plus complexes dans le cas des entiers que dans le cas des polynômes, à cause des retenues, et nous n'en détaillons donc que la version polynomiale. Dans les définitions et résultats de nature moins algorithmique, nous utilisons le cadre algébrique des anneaux euclidiens qui permet de traiter simultanément toutes les applications que nous avons en vue. L'algorithme d'Euclide pour le calcul du pgcd est probablement le plus vieil algorithme non-trivial encore en utilisation. Il est présenté en Section 6.1. Dans le cas des polynômes de $\mathbb{K}[X]$, sa complexité est quadratique en nombre d'opérations dans le corps $\mathbb{K}$. Le résultant est également lié à l'algorithme d'Euclide, ses propriétés et son calcul sont présentés en Section 6.2. En outre, une technique dite « des sous-résultants » permet de réduire l'explosion de la taille des coefficients intermédiaires dont souffre l'algorithme d'Euclide pour le pgcd dans $\mathbb{Q}[X]$. Le chapitre se termine en Section 6.3 sur un algorithme de complexité quasi-optimale pour le pgcd dans $\mathbb{K}[X]$, exploitant la multiplication polynomiale rapide.

Dans ce chapitre, $\mathbb{A}$ désignera toujours un anneau intègre.

6.1 Algorithme d'Euclide

Le pgcd

Dans tout anneau intègre $\mathbb{A}$, on appelle *plus grand diviseur commun* (pgcd) de $A \in \mathbb{A}$ et de $B \in \mathbb{A}$ tout $G \in \mathbb{A}$ qui divise A et B et tel que tout diviseur de A et de B divise

aussi G. Cependant, dans ce degré de généralité, la notion est délicate : il existe des anneaux intègres contenant des éléments sans pgcd.

Exercice 6.1 Montrer que dans $\mathbb{A} = \mathbb{Z}[i\sqrt{3}]$, les éléments $A = 4$ et $B = 2 + 2i\sqrt{3}$ admettent $G_1 = 2$ et $G_2 = 1 + i\sqrt{3}$ comme diviseurs, mais n'admettent aucun pgcd. ■

On appelle traditionnellement *anneau à pgcd* un anneau intègre dans lequel tous couple d'éléments admet un pgcd. Algébriquement, le bon cadre pour parler de pgcd est celui des anneaux factoriels ; d'un point de vue algorithmique, une classe d'anneaux à pgcd bien adaptée aux calculs est celle des anneaux euclidiens. Contrairement à l'usage, nous notons dans ce chapitre pgcd(A, B) tout pgcd de A et de B sans faire de choix de normalisation parmi les pgcd de A et de B. L'algorithme d'Euclide présenté dans cette section permet le calcul du pgcd dans $\mathbb{Z}$ ou dans l'anneau $\mathbb{K}[X]$, où $\mathbb{K}$ est un corps. Il repose sur l'existence d'une *division euclidienne* dans ces anneaux. Un anneau intègre $\mathbb{A}$ est appelé *anneau euclidien* s'il existe une fonction de taille $d : \mathbb{A} \to \mathbb{N} \cup \{-\infty\}$ telle que pour tout $a \in \mathbb{A}$, $b \in \mathbb{A} \setminus \{0\}$, il existe q et r dans $\mathbb{A}$ avec

$$a = qb + r, \qquad d(r) < d(b).$$

En particulier, $d(0) < d(b)$ dès que b est non nul. Dans le cas des entiers, la valeur absolue fournit une telle fonction d, et le degré remplit ce rôle pour les polynômes. La notation $r := a \bmod b$ sert dans les deux cas à définir r à partir de a et de b. L'entier ou le polynôme r est appelé *reste* de la division euclidienne de a par b.

R Un élément d'un anneau est dit irréductible si les produits qui lui sont égaux font tous intervenir un élément inversible. Tout anneau euclidien est *principal* (c'est-à-dire composé des multiples d'un élément donné de l'anneau), et en particulier il est *factoriel* (c'est-à-dire que tout élément non nul y a une factorisation unique en irréductibles, à l'ordre près de la factorisation et à des multiplications près des facteurs par des inversibles de l'anneau). Le pgcd se lit aussi sur les factorisations. Si deux éléments A et B d'un anneau factoriel $\mathbb{A}$ se factorisent sous la forme

$$A = a m_1^{k_1} \cdots m_s^{k_s}, \qquad B = b m_1^{\ell_1} \cdots m_s^{\ell_s},$$

où a et b sont inversibles dans $\mathbb{A}$, les m_i sont irréductibles et les k_i et ℓ_i sont des entiers positifs ou nuls, montrer qu'alors un pgcd de A et B est

$$G = m_1^{\min(k_1,\ell_1)} \cdots m_s^{\min(k_s,\ell_s)}.$$

Autrement dit, on rassemble dans G les facteurs irréductibles communs de A et B, avec la plus grande multiplicité possible. La factorisation dans $\mathbb{Z}$ ou dans $\mathbb{K}[X]$ est plus coûteuse que le pgcd et cette propriété n'est donc pas utilisée pour le calcul du pgcd.

Calcul du pgcd

Étant donnés A et B dans un anneau euclidien $\mathbb{A}$, l'Algorithme 6.1 d'Euclide calcule une suite de restes successifs dont la taille décroît, jusqu'à atteindre le pgcd. La terminaison provient de la décroissance stricte de la taille à chaque étape. La correction de cet algorithme se déduit de la relation

$$\text{pgcd}(F, G) = \text{pgcd}(H, G) \qquad \text{pour} \qquad H := F \bmod G.$$

Entrée A et B dans $\mathbb{A}$.
Sortie Un pgcd de A et B.
1. $R_0 := A$; $R_1 := B$; $i := 1$.
2. Tant que R_i est non nul, faire :
   $R_{i+1} := R_{i-1} \bmod R_i$
   $i := i + 1$.
3. Renvoyer R_{i-1}.

Algorithme 6.1 – Algorithme d'Euclide.

Par récurrence, il s'ensuit que $\operatorname{pgcd}(F, G) = \operatorname{pgcd}(R_i, R_{i+1})$ pour tout i. Si en outre R_{i+1} est nul, alors $\operatorname{pgcd}(R_i, R_{i+1}) = R_i$, ce qui prouve la correction.

Exemple 6.1 Soient $A = X^4 - 13X^3 + 2X^2 - X - 1$ et $B = X^2 - X - 1$ dans $\mathbb{Q}[X]$. La suite des restes est :

$$R_0 = X^4 - 13X^3 + 2X^2 - X - 1, \quad R_1 = X^2 - X - 1,$$
$$R_2 = -22X - 10, \quad R_3 = -41/121, \quad R_4 = 0,$$

de sorte que $-41/121$ est un pgcd de A et B et donc 1 aussi.

Théorème 6.1 L'algorithme d'Euclide calcule un pgcd de deux polynômes A et B dans $\mathbb{K}[X]$ en $O(\deg A \deg B)$ opérations dans $\mathbb{K}$.

Démonstration. La correction a été prouvée dans le cas général. Pour l'étude de complexité, nous supposons d'abord que $\deg A \geq \deg B$. D'après le Chapitre 3 (page 82), le calcul naïf de P mod Q peut être effectué en $(2\deg Q + 1)(\deg P - \deg Q + 1) + 1$ opérations de $\mathbb{K}$. Il s'ensuit que le coût de l'algorithme d'Euclide est borné par la somme des $(2\deg(R_i) + 1)(\deg R_{i-1} - \deg R_i + 1) + 1$, pour $i \geq 1$. Tous les $\deg(R_i)$ pour $i \geq 1$ sont majorés par $\deg B$, de sorte que le coût est borné par $(2\deg B + 1)\sum_{i\geq 1}(\deg R_{i-1} - \deg R_i + 1) + \deg B \leq (2\deg B + 2)(\deg R_0 + \deg B) = O(\deg A \deg B)$.

Si le degré de B est supérieur à celui de A, la première étape ne coûte pas d'opération arithmétique, et la borne ci-dessus s'applique au reste du calcul. ■

Exercice 6.2 Énoncer et prouver l'analogue entier du résultat du Théorème 6.1. ■

La borne de complexité quadratique reflète bien le comportement de l'algorithme : dans une exécution typique les quotients ont degré 1 à chaque itération, les degrés des restes successifs diminuent de 1 à chaque itération et leur calcul est linéaire ; le nombre de coefficients intermédiaires calculés est quadratique et ils sont calculés aussi efficacement que possible par un algorithme qui les calcule tous.

Pgcd étendu et inversion modulaire

Relation de Bézout

Une relation de la forme $G = UA + VB$ qui donne le pgcd G de deux éléments A et B de $\mathbb{A}$ avec deux cofacteurs U et V dans $\mathbb{A}$ est appelée une *relation de Bézout*. L'algorithme d'Euclide étendu est une modification légère de l'algorithme d'Euclide qui calcule non seulement le pgcd, mais aussi une relation de Bézout particulière,

$$UA + VB = G, \quad \text{avec } d(UG) < d(B) \text{ et } d(VG) < d(A). \tag{6.1}$$

Une fois le pgcd G choisi, les cofacteurs U et V sont rendus uniques par la contrainte sur les tailles. Dans de nombreuses applications, c'est davantage de ces *cofacteurs* U et V que du pgcd lui-même dont on a besoin. On parle alors de calcul de pgcd *étendu*. Ce calcul intervient par exemple de manière importante dans la manipulation des nombres algébriques (Exemple 6.3 ci-dessous), dans le calcul dans des algèbres quotient $\mathbb{K}[X]/(P)$ (Exemples 6.2 et 6.5 ci-dessous, cf. aussi la Section 4.2 en page 84), et dans le développement rapide des séries algébriques (Chapitre 14). Il intervient également dans d'autres algorithmes qui ne seront pas abordés dans cet ouvrage, par exemple pour le calcul des « restes chinois rapides », pour la factorisation de polynômes dans $\mathbb{Z}[X]$ ou $\mathbb{Q}[X]$ par des techniques de type « Hensel » (Chapitre 21), ou pour l'intégration symbolique (à la Hermite).

Mais, avant tout, les calculs de pgcd étendu permettent d'effectuer des *inversions modulaires*. Lorsque A et B sont premiers entre eux, (c'est-à-dire si G est un élément inversible de A), alors l'élément V est un inverse de B modulo A.

Exemple 6.2 La relation de Bézout pour $a + bX$ et $1 + X^2$ dans $\mathbb{R}[X]$ s'écrit :

$$(a - bX)(a + bX) + b^2(1 + X^2) = a^2 + b^2.$$

L'inverse de $B = a + bX$ modulo $A = 1 + X^2$ vaut donc

$$V = \frac{a - bX}{a^2 + b^2}.$$

Puisque le corps des complexes s'obtient par le quotient $\mathbb{R}[X]/(X^2+1)$, cette relation n'est autre que la formule d'inversion familière

$$\frac{1}{a + ib} = \frac{a - ib}{a^2 + b^2} \qquad \text{où } i = \sqrt{-1}.$$

Si $A \in \mathbb{A}$ est irréductible, alors $\mathbb{A}/(A)$ est un corps, et le calcul de la relation de Bézout permet d'y effectuer la division : si $B \neq 0 \bmod A$, alors $N/B = NV \bmod A$.

Exemple 6.3 La « simplification » de la fraction rationnelle

$$r = \frac{\phi^4 - \phi + 1}{\phi^7 - 1} = \frac{25\phi - 34}{29}$$

où ϕ est le « nombre d'or », défini par $A(\phi) = 0$ pour $A(X) = X^2 - X - 1$, est obtenue par les trois calculs suivants :
- calcul du reste $U = 13X + 7$ de la division euclidienne de $X^7 - 1$ par A,
- détermination de la relation de Bézout $(13X - 20)U - 169A = 29$,
- calcul du reste $V = 25X - 34$ de la division euclidienne de $(13X - 20)(X^4 - X + 1)$ par A.

Plus généralement, dans le cas où $A \in \mathbb{K}[X]$ est un polynôme irréductible, ces calculs montrent que le corps $\mathbb{K}(\alpha)$ des fractions rationnelles en α racine de A est un espace vectoriel dont une base est $1, \alpha, \alpha^2, \ldots, \alpha^{\deg A - 1}$. Les algorithmes d'Euclide et d'Euclide étendu fournissent un moyen de calcul dans cette représentation. Il est ainsi possible de manipuler de manière exacte les racines d'un polynôme de degré arbitraire sans « résoudre ».

Exemple 6.4 Le même calcul qu'à l'Exemple 6.3 fonctionne sur des entiers :

$$r = \frac{25}{33} \equiv 5 \bmod 7$$

se déduit de $33 \bmod 7 = 5$, $3 \times 5 - 2 \times 7 = 1$, $3 \times 25 \bmod 7 = 5$.

Exemple 6.5 Lorsque l'élément $A \in \mathbb{A}$ n'est pas irréductible, $\mathbb{A}/(A)$ n'est pas un corps. Cependant, les éléments inversibles peuvent y être inversés par le même calcul que ci-dessus. Lorsqu'un élément B non nul n'est pas inversible, la relation de Bézout se produit avec un G différent de 1. Il est alors possible de tirer parti de cette information (un facteur de A) en scindant le calcul d'une part sur G et d'autre part sur B/G.

Exercice 6.3 Soit p un nombre premier et soit a un entier non multiple de p. Donner un algorithme pour le calcul de l'inverse de a modulo p utilisant le petit théorème de Fermat ($a^{p-1} = 1 \bmod p$) et l'exponentiation binaire. Analyser la complexité binaire de cet algorithme et comparer avec l'approche par Euclide étendu. ■

Calcul des cofacteurs

L'idée-clé est de suivre pendant l'algorithme d'Euclide la décomposition de chacun des restes R_i sur A et B. Autrement dit, pour tout i, l'algorithme calcule des éléments U_i et V_i tels que

$$U_i A + V_i B = R_i,$$

dont les tailles sont bien contrôlées. Pour $i = 0$, il suffit de poser $U_0 = 1, V_0 = 0$, ce qui correspond à l'égalité $1 \cdot A + 0 \cdot B = A = R_0$. Pour $i = 1$, l'égalité $0 \cdot A + 1 \cdot B = B = R_1$ est obtenue avec $U_1 = 0, V_1 = 1$. Ensuite, la division euclidienne $R_{i-1} = Q_i R_i + R_{i+1}$ donne la relation

$$R_{i+1} = R_{i-1} - Q_i R_i = (U_{i-1} - Q_i U_i)A + (V_{i-1} - Q_i V_i)B,$$

qui pousse à définir U_{i+1} par $U_{i-1} - Q_i U_i$ et V_{i+1} par $V_{i-1} - Q_i V_i$. À partir de cette relation, une preuve par récurrence montre que les conditions de tailles de la relation

Entrée A et B dans $\mathbb{A}$.
Sortie Un pgcd G de A et B, et les cofacteurs U et V de l'identité de Bézout $UA + VB = G$.

1. $R_0 := A$; $U_0 := 1$; $V_0 := 0$; $R_1 := B$; $U_1 := 0$; $V_1 := 1$; $i := 1$.
2. Tant que R_i est non nul, faire :
   a. Calculer le quotient Q_i et le reste R_{i+1} de la division euclidienne de R_{i-1} par R_i.
   b. $U_{i+1} := U_{i-1} - Q_i U_i$; $V_{i+1} := V_{i-1} - Q_i V_i$.
   c. $i := i + 1$.
3. Renvoyer $R_{i-1}, U_{i-1}, V_{i-1}$.

Algorithme 6.2 – Algorithme d'Euclide étendu.

de Bézout sont satisfaites. La méthode est résumée dans l'Algorithme 6.2. À nouveau, dans le cas des polynômes ou des entiers, la complexité est quadratique.

6.2 Résultant

Matrice de Sylvester

L'algorithme d'Euclide pour *deux* polynômes A, B à *une* variable est étroitement relié à la matrice de Sylvester. Si

$$A = a_m X^m + \cdots + a_0, \quad (a_m \neq 0), \qquad \text{et} \qquad B = b_n X^n + \cdots + b_0, \quad (b_n \neq 0),$$

appartiennent à $\mathbb{A}[X]$, cette matrice, carrée de taille $m+n$, s'écrit

$$\mathrm{Syl}(A,B) = \begin{pmatrix} a_m & \dots & a_1 & a_0 & & \\ & \ddots & & & \ddots & \\ & & a_m & a_{m-1} & \dots & a_0 \\ b_n & \dots & b_1 & b_0 & & \\ & \ddots & & & \ddots & \\ & & b_n & b_{n-1} & \dots & b_0 \end{pmatrix}$$

avec les n premières lignes contenant les coefficients de A et les m suivantes contenant les coefficients de B. La transposée de cette matrice représente l'application linéaire

$$\varphi_{A,B} : (U, V) \mapsto AU + BV,$$

en se limitant aux paires (U, V) telles que $\deg U < \deg B$ et $\deg V < \deg A$, et en utilisant la base $(X^{n-1}, 0), (X^{n-2}, 0), \dots (1, 0), (0, X^{m-1}), (0, X^{m-2}), \dots (0, 1)$ au départ et la base $(X^{m+n-1}, X^{m+n-2}, \dots, 1)$ à l'arrivée. Les combinaisons linéaires des lignes de la matrice de Sylvester donnent donc les coefficients des polynômes qui peuvent être obtenus comme image.

Lorsque $\mathbb{A}$ est un corps, au vu de la relation de Bézout (6.1), le pgcd de A et B peut être obtenu ainsi et il est caractérisé comme l'élément unitaire de plus petit

degré qui peut l'être. Ceci permet de le lire sur une forme *échelonnée en ligne* de la matrice Syl(A, B) (cf. Définition 8.3, page 170). Qui plus est, le rang de Syl(A, B) vaut la dimension de l'image de l'application $\varphi_{A,B}$, qui est égale à $m+n-\deg \operatorname{pgcd}(A,B)$. Par le théorème du rang, il s'ensuit que le degré du pgcd(A, B) est égal à la dimension du noyau de Syl(A, B), c'est-à-dire au nombre de lignes nulles de sa forme échelonnée.

Proposition 6.2 Soient A, B deux polynômes de $\mathbb{K}[X]$ où $\mathbb{K}$ est un corps. La forme échelonnée en ligne de la matrice de Sylvester Syl(A, B) contient sur sa dernière ligne non nulle les coefficients d'un pgcd de A et B. En outre, le nombre de lignes nulles de cette matrice échelonnée est le degré du pgcd de A et de B.

Exemple 6.6 Pour les polynômes $A = X^4 - X^3 - 7X^2 + 2X + 3$ et $B = X^3 - 4X^2 + 2X + 3$, la matrice de Sylvester et sa forme échelonnée en ligne sont :

$$\begin{pmatrix} 1 & -1 & -7 & 2 & 3 & 0 & 0 \\ 0 & 1 & -1 & -7 & 2 & 3 & 0 \\ 0 & 0 & 1 & -1 & -7 & 2 & 3 \\ 1 & -4 & 2 & 3 & 0 & 0 & 0 \\ 0 & 1 & -4 & 2 & 3 & 0 & 0 \\ 0 & 0 & 1 & -4 & 2 & 3 & 0 \\ 0 & 0 & 0 & 1 & -4 & 2 & 3 \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} 1 & -1 & -7 & 2 & 3 & 0 & 0 \\ 0 & 1 & -1 & -7 & 2 & 3 & 0 \\ 0 & 0 & 1 & -1 & -7 & 2 & 3 \\ 0 & 0 & 0 & -14 & 45 & -3 & -18 \\ 0 & 0 & 0 & 0 & -\frac{5}{7} & \frac{12}{7} & \frac{9}{7} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{10} & -\frac{3}{10} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

La dernière ligne non nulle donne le pgcd $X - 3$.

Définition 6.1 Le *résultant* de A et B est le déterminant de la matrice de Sylvester Syl(A, B). Il est noté Res(A, B), ou $\operatorname{Res}_X(A,B)$ si l'on veut insister sur l'élimination de la variable X.

L'utilisation des résultants pour l'élimination demande de travailler dans ce chapitre avec des matrices à coefficients dans un anneau commutatif qui n'est pas nécessairement un corps. Les définitions et propriétés du déterminant persistent dans ce cas (c'est une forme multilinéaire alternée, le déterminant de la matrice identité vaut 1, on peut le développer par rapport à une ligne ou une colonne, il ne change pas par addition de combinaisons linéaires de lignes ou de colonnes, et s'exprime comme somme sur les permutations).

Le résultant peut être vu comme une condition de cohérence pour le système formé par les deux polynômes A et B.

Corollaire 6.3 Soient A et B deux polynômes de $\mathbb{K}[X]$. Alors A et B sont premiers entre eux si et seulement si $\operatorname{Res}(A,B) \neq 0$.

Démonstration. Le déterminant d'une matrice carrée se lit sur la forme échelonnée en ligne en faisant le produit des éléments diagonaux. Il est non nul si et seulement si tous les éléments diagonaux le sont et dans ce cas un pgcd non nul est sur la dernière ligne. À l'inverse, s'il existe un pgcd G non nul, alors les polynômes $X^k G$ montrent

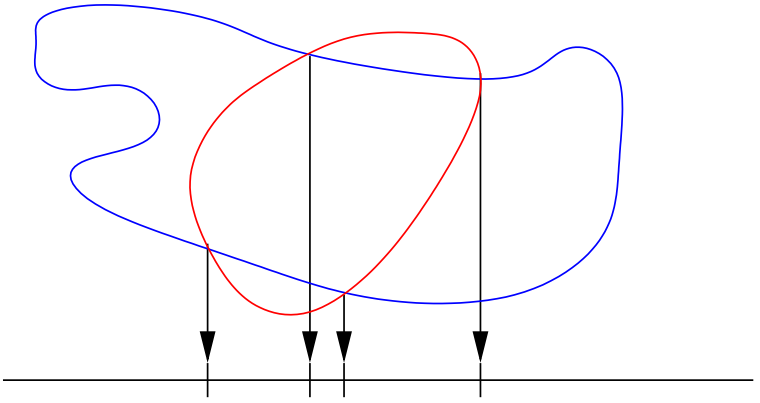


FIGURE 6.3 – Le résultant calcule des projections.

l'existence de lignes avec un coefficient de tête non nul dans chaque colonne de la forme échelonnée. ■

Exemple 6.7 Le discriminant d'un polynôme $A = a_m X^m + \cdots + a_1 X + a_0$ de degré m à coefficients dans un corps $\mathbb{K}$ est le polynôme $\mathrm{disc}(A)$ défini par

$$\mathrm{Res}(A, A') = (-1)^{m(m-1)/2} a_m \,\mathrm{disc}(A).$$

Il s'annule lorsque ces deux polynômes ont une racine commune dans une clôture algébrique de $\mathbb{K}$, c'est-à-dire, en caractéristique nulle, lorsque A a une racine multiple.

Exercice 6.4 Calculer le discriminant de $aX^2 + bX + c$ en prenant le déterminant de la matrice de Sylvester. ■

Applications du résultant

Calculs de projections

Algébriquement, la « résolution » d'un système polynomial se ramène souvent à une question d'*élimination*. Lorsqu'elle est possible, l'élimination successive des variables amène le système d'entrée sous une forme triangulaire. De proche en proche, la résolution d'un tel système se réduit alors à la manipulation de polynômes à une variable, pour lesquels compter, isoler, … , les solutions est bien plus facile. Géométriquement, l'élimination correspond à une projection. Dans cette section nous montrons comment le résultant de deux polynômes permet de traiter ce genre de problèmes, pour les systèmes de deux équations en deux inconnues. Le schéma général est représenté en Figure 6.3. Cette opération est à la base d'une technique plus générale pour un nombre arbitraire d'équations et d'inconnues, la *résolution géométrique*, qui sera présentée au Chapitre 28.

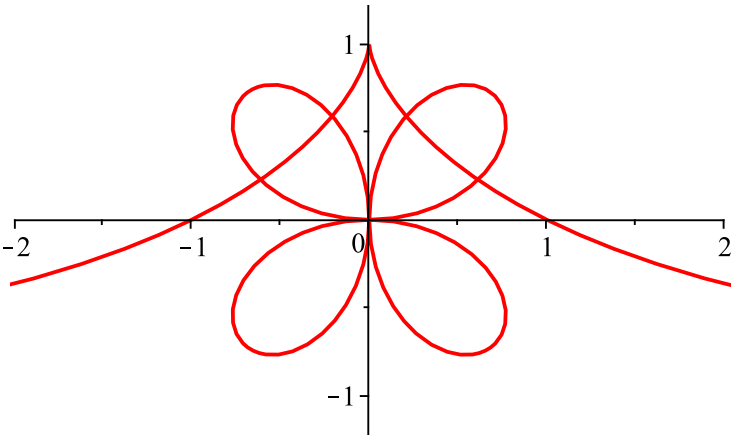


FIGURE 6.4 – Courbes de l'Exemple 6.8.

Exemple 6.8 Les deux courbes de la Figure 6.4 ont pour équations

$$A = (X^2 + Y^2)^3 - 4X^2Y^2 = 0, \qquad B = X^2(1+Y) - (1-Y)^3 = 0.$$

Ces deux polynômes sont irréductibles et leur pgcd, qui vaut 1, ne renseigne pas sur leurs racines communes. Les résultants par rapport à X et Y donnent en revanche des polynômes qui s'annulent sur les coordonnées de ces points d'intersection :

$$\mathrm{Res}_X(A,B) = (4Y^7 + 60Y^6 - 152Y^5 + 164Y^4 - 95Y^3 + 35Y^2 - 9Y + 1)^2,$$
$$\mathrm{Res}_Y(A,B) = 16X^{14} + 6032X^{12} - 1624X^{10} + 4192X^8 - 815X^6 - 301X^4 - 9X^2 + 1.$$

Il n'y a que 4 points d'intersection visibles sur la Figure 6.4, les 10 autres ont au moins une coordonnée qui n'est pas réelle. Le caractère bicarré du résultant en Y provient de la symétrie de la figure par rapport à l'axe des Y. C'est pour cette même raison que le premier résultant est un carré.

Le résultat général est le suivant.

Proposition 6.4 Soient $A = a_m Y^m + \cdots$ et $B = b_n Y^n + \cdots$ où les coefficients a_i et b_j sont dans $\mathbb{K}[X]$ où $\mathbb{K}$ est un corps algébriquement clos. Alors les racines du polynôme $\mathrm{Res}_Y(A,B) \in \mathbb{K}[X]$ sont d'une part les abscisses des solutions du système $A = B = 0$, d'autre part les racines des termes de tête a_m et b_n.

Démonstration. La preuve repose sur des résultats de la section suivante, qui sont indépendants de cette proposition.

Tout d'abord, les racines des termes de tête a_m et b_n sont racines du résultant d'après la formule de Poisson (Théorème 6.5 ci-dessous).

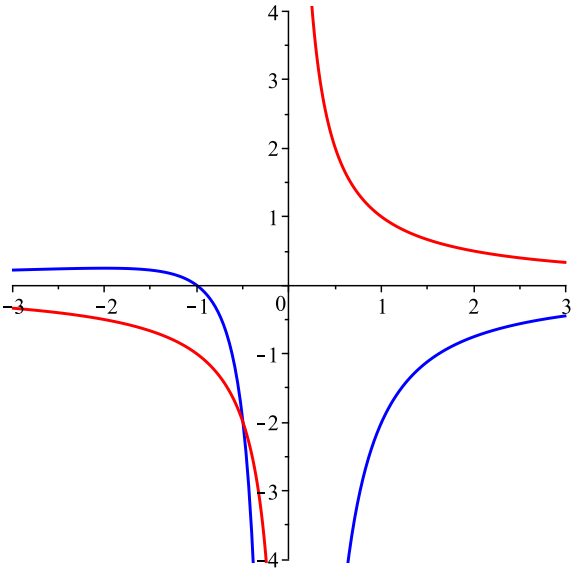


FIGURE 6.5 – Courbes de l'Exemple 6.9.

Ensuite, d'après la Proposition 6.7 ci-dessous, il existe U et V dans $\mathbb{K}[X,Y]$ tels que

$$\mathrm{Res}_Y(A,B) = UA + VB.$$

En évaluant cette identité en (x,y) tel que $A(x,y) = B(x,y) = 0$, on voit qu'un tel x est racine de $\mathrm{Res}_Y(A,B)$. À l'inverse, si x annule le résultant, alors d'après le Corollaire 6.3, les polynômes $A(x,Y)$ et $B(x,Y)$ ont un pgcd de degré au moins 1. Comme le corps est algébriquement clos, il existe alors $y \in \mathbb{K}$ racine de ce pgcd tel que $A(x,y) = B(x,y) = 0$. ■

Graphiquement, les racines « dégénérées » du second cas se traduisent par la présence d'asymptotes verticales.

Exemple 6.9 Avec $A = X^2Y + X + 1$, $B = XY - 1$, on obtient $\mathrm{Res}_Y(A,B) = -X(2X+1)$ (asymptote en $X = 0$, « vraie » solution en $X = -\frac{1}{2}$). Les courbes correspondantes sont représentées en Figure 6.5 : celle pour A en bleu, celle pour B en rouge.

Une fois calculé, le résultant donnant les coordonnées des abscisses des intersections des deux courbes, il est souhaitable d'obtenir les ordonnées correspondantes. Dans le cas simple où l'avant-dernier reste de l'algorithme d'Euclide est de degré 1, il fournit bien une telle paramétrisation.

Exemple 6.10 Sur les deux polynômes A et B de l'Exemple 6.8, vus comme poly-

nômes dans $\mathbb{Q}(X)[Y]$, la suite des restes est

$$(X^4 - 13X^2 + 15)Y^2 - 4(X^2 + 2)(X^2 + 3)Y + (X^6 - 2X^4 - 8X^2 + 10),$$
$$(4X^8 - 344X^6 - 1243X^4 - 301X^2 - 21)Y + (84X^8 - 372X^6 + 169X^4 + 143X^2 + 15),$$
$$1.$$

Un calcul de pgcd du coefficient de Y dans l'avant-dernier reste avec $\mathrm{Res}_Y(A, B)$ montre que ces deux polynômes sont premiers entre eux. Pour chaque racine X de $\mathrm{Res}_Y(A, B)$ il y a donc un unique point d'intersection aux deux courbes, d'ordonnée donnée par

$$Y = -\frac{84X^8 - 372X^6 + 169X^4 + 143X^2 + 15}{4X^8 - 344X^6 - 1243X^4 - 301X^2 - 21}.$$

En utilisant un calcul de pgcd étendu, ceci peut être récrit, comme expliqué plus haut, en un polynôme de degré au plus 13 en X.

Le même calcul sur A et B vus comme polynômes dans $\mathbb{Q}(Y)[X]$ donne B comme dernier reste avant le pgcd. Ceci correspond aux deux points ayant la même projection sur l'axe des Y.

Implicitation

Une autre application géométrique du résultant est l'*implicitation* de courbes du plan. Étant donnée une courbe paramétrée

$$X = A(T), \quad Y = B(T), \qquad A, B \in \mathbb{K}(T),$$

il s'agit de calculer un polynôme non trivial en X et Y qui s'annule sur la courbe. Il suffit pour cela de prendre le résultant en T des numérateurs de $X - A(T)$ et de $Y - B(T)$.

Exemple 6.11 La courbe « en fleur » de la Figure 6.4, plus savamment appelée « quadrifolium », peut aussi être donnée sous la forme

$$X = \frac{4T(1 - T^2)^2}{(1 + T^2)^3}, \quad Y = \frac{8T^2(1 - T^2)}{(1 + T^2)^3}.$$

Il suffit d'effectuer le calcul du résultant

$$\mathrm{Res}_T((1 + T^2)^3 X - 4T(1 - T^2)^2, (1 + T^2)^3 Y - 8T^2(1 - T^2))$$

pour retrouver (à un facteur constant près) le polynôme A de l'Exemple 6.8.

Propriétés

L'essentiel des propriétés du résultant est fourni par le théorème suivant.

Théorème 6.5 — Formule de Poisson. Si les polynômes A et B de $\mathbb{A}[X]$ s'écrivent

$$A = a(X-\alpha_1)\cdots(X-\alpha_m), \quad B = b(X-\beta_1)\cdots(X-\beta_n),$$

alors, le résultant de A et B vaut

$$\begin{aligned}\operatorname{Res}(A,B) &= a^n b^m \prod_{i,j}(\alpha_i-\beta_j) = (-1)^{mn} b^m \prod_{1\le j\le n} A(\beta_j)\\ &= a^n \prod_{1\le i\le m} B(\alpha_i) = (-1)^{mn}\operatorname{Res}(B,A).\end{aligned}$$

Démonstration. Il suffit de prouver la première égalité. Les trois suivantes en sont des conséquences immédiates.

Le facteur $a^n b^m$ est une conséquence de la multilinéarité du déterminant. On peut donc se restreindre au cas où $a = b = 1$. Il est commode de considérer le cas générique où les α_i et β_j sont des indéterminées et où l'anneau $\mathbb{A}$ est $\mathbb{Z}[\alpha_1,\dots,\alpha_m,\beta_1,\dots,\beta_n]$. Si le résultat, qui est une identité entre polynômes, est vrai dans cet anneau, il l'est aussi par spécialisation pour des valeurs arbitraires des α_i et β_j. Le Corollaire 6.3 avec $\mathbb{K} = \mathbb{Q}(\alpha_1,\dots,\alpha_m,\beta_1,\dots,\beta_n)$ montre que le polynôme obtenu en remplaçant la variable β_j par la variable α_i dans le résultant est le polynôme identiquement nul. Il s'ensuit que le produit des $\alpha_i-\beta_j$ divise le résultant. Par ailleurs, le degré en α_i de chacune des n premières lignes de la matrice de Sylvester est 1, et ce degré est nul pour les m autres lignes. Ceci donne une borne n pour le degré en chaque α_i du résultant et de la même manière une borne m pour son degré en chaque β_j. Il s'ensuit que le résultant est égal au produit des $\alpha_i-\beta_j$ à un facteur constant près. Ce facteur est indépendant des α_i et β_j. En choisissant $\alpha_1 = \cdots = \alpha_m = 0$, c'est-à-dire $A = X^m$, la matrice de Sylvester est triangulaire, et son déterminant vaut $B(0)^m$, ce qui donne le facteur 1 et conclut la preuve. ■

Exemple 6.12 Si $A = a\prod_{i=1}^m (X-x_i)$, la formule de Poisson entraîne que le discriminant $\operatorname{disc}(A)$ de A est égal à :

$$(-1)^{\frac{m(m-1)}{2}} a^{m-2} \prod_{i=1}^m A'(x_i) = (-1)^{\frac{m(m-1)}{2}} a^{m-2} \prod_{i=1}^m a\prod_{j\neq i}(x_i-x_j) = a^{2m-2}\prod_{i<j}(x_i-x_j)^2.$$

Corollaire 6.6 — Multiplicativité du résultant. Pour tous polynômes A, B, C de $\mathbb{A}[X]$, $\operatorname{Res}(AB,C) = \operatorname{Res}(A,C)\operatorname{Res}(B,C)$.

Démonstration. L'anneau $\mathbb{A}$ étant intègre, il possède un corps de fractions qui a lui-même une clôture algébrique dans laquelle les polynômes A, B et C peuvent s'écrire sous la forme utilisée dans le théorème précédent. L'identité sur les résultants (qui appartiennent à $\mathbb{A}$ par définition) est alors vérifiée en considérant les produits deux à deux de racines. ■

Une dernière propriété utile du résultant est la suivante.

Proposition 6.7 Il existe U et V dans $\mathbb{A}[X]$ avec $\deg U < \deg B$ et $\deg V < \deg A$, tels que le résultant de A et de B s'écrive $\mathrm{Res}(A,B) = UA + VB$.

Démonstration. En ajoutant à la dernière colonne de la matrice de Sylvester le produit des colonnes précédentes par des puissances adaptées de X, on fait apparaître dans la dernière colonne les polynômes $X^{\deg B-1}A, \ldots, XA, A$ et $X^{\deg A-1}B, \ldots, XB, B$, sans avoir changé le déterminant. Le développement du déterminant par rapport à la dernière colonne permet alors de conclure. ■

L'algorithme d'Euclide étendu montre que le résultant $\mathrm{Res}(A,B)$ (qui est un multiple par un élément du corps des fractions $\mathbb{K}$ de l'un des restes euclidiens « standards »), peut s'écrire comme une combinaison polynomiale de A et B, à coefficients dans $\mathbb{K}[X]$. La Proposition 6.7 montre que cette combinaison se fait « sans division », c'est-à-dire avec des coefficients dans $\mathbb{A}[X]$.

Spécialisation du résultant

Spécialisation et résultant ne commutent pas toujours, comme le montre l'exemple suivant.

Exemple 6.13 Le résultant de $aX^2 + bX + c$ et $dX + e$ vaut $ae^2 + cd^2 - bde$, qui vaut $d(cd - be)$ en $a = 0$, alors que le résultant de $bX + c$ et $dX + e$ vaut $be - cd$.

Le résultat suivant fournit une condition simple pour que le résultant de la spécialisation de deux polynômes soit égal à la spécialisation de leur résultant, propriété qui sera utile pour les calculs sur les systèmes polynomiaux de la Partie V.

Proposition 6.8 Soient $\mathbb{A}$ et $\mathbb{B}$ deux anneaux commutatifs avec des unités respectives $1_{\mathbb{A}}$ et $1_{\mathbb{B}}$. Soit φ un morphisme de $\mathbb{A}$ dans $\mathbb{B}$ qui envoie $1_{\mathbb{A}}$ sur $1_{\mathbb{B}}$, et que l'on étend naturellement à $\mathbb{A}[X]$ en l'appliquant coefficient par coefficient. Soient A et B deux polynômes de $\mathbb{A}[X]$. Si A est unitaire alors $\varphi(\mathrm{Res}(A,B)) = \mathrm{Res}(\varphi(A), \varphi(B))$.

Démonstration. Le lemme ci-dessous donne une expression du résultant comme le déterminant de l'endomorphisme de multiplication par B dans $\mathbb{A}[X]/(A(X))$, de matrice notée $M(A,B)$ dans la base $1, X, \ldots, X^{\deg(A)-1}$. Sa i-ième colonne est le reste R_i de la division euclidienne de X^iB par A. Comme A est unitaire, les coefficients de R_i sont des polynômes en les coefficients de A et B. Par ailleurs, le reste de la division euclidienne de $\varphi(X^iB)$ par $\varphi(A)$ coïncide avec $\varphi(R_i)$. On en déduit que $\varphi(M(A,B)) = M(\varphi(A), \varphi(B))$. La conclusion vient du fait que $\varphi(\det(M(A,B)) = \det(\varphi(M(A,B)))$. ■

Lemme 6.9 Soit $\mathbb{A}$ un anneau commutatif avec une unité notée $1_{\mathbb{A}}$, et soient A et B deux polynômes de $\mathbb{A}[X]$. Si A est unitaire, alors le résultant $\mathrm{Res}(A,B)$ est égal au déterminant de l'endomorphisme de multiplication par B dans $\mathbb{A}[X]/(A(X))$.

Démonstration. Notons $A = 1_{\mathbb{A}}X^m + a_{m-1}X^{m-1} + \cdots + a_0$, et $B = b_nX^n + \cdots + b_0$, avec $b_n \neq 0$. La matrice de Sylvester Syl(A, B) s'écrit :

$$\mathrm{Syl}(A,B) = \begin{pmatrix} 1_{\mathbb{A}} & \dots & a_1 & a_0 & & \\ & \ddots & & & \ddots & \\ & & 1_{\mathbb{A}} & a_{m-1} & \dots & a_0 \\ b_n & \dots & b_1 & b_0 & & \\ & \ddots & & & \ddots & \\ & & b_n & b_{n-1} & \dots & b_0 \end{pmatrix}$$

En notant $R_i = r_{i,m-1}X^{m-1} + \cdots + r_{i,0}$ le reste de la division euclidienne de X^iB par A, et en effectuant les opérations sur les lignes de Syl(A, B) correspondant à ces divisions, on constate que le déterminant de cette matrice est égal au déterminant de la matrice

$$\begin{pmatrix} r_{m-1,m-1} & r_{m-1,m-2} & \cdots & r_{m-1,0} \\ r_{m-2,m-1} & r_{m-2,m-2} & \cdots & r_{m-2,0} \\ \vdots & \vdots & \dots & \vdots \\ r_{0,m-1} & r_{0,m-2} & \cdots & r_{0,0} \end{pmatrix},$$

dont la transposée n'est autre que la matrice de multiplication par B dans $\mathbb{A}[X]/(A(X))$, pour la base $X^{m-1}, \dots, X, 1$. ■

Calcul en coût quadratique

En utilisant la Définition 6.1 et les résultats du Chapitre 8, le résultant de deux polynômes A, B de $\mathbb{K}[X]$ peut se calculer naïvement en $O(\mathsf{MM}(n)) = O(n^\theta)$ opérations dans $\mathbb{K}$, où $n = \max(\deg(A), \deg(B))$ et où $2 \leq \theta \leq 3$ est un exposant réalisable pour la multiplication matricielle. Un algorithme plus efficace, de complexité seulement quadratique, s'obtient comme conséquence du Théorème 6.5 et est laissé en exercice.

Exercice 6.5 — Algorithme de type Euclide pour le résultant. Soient A et B deux polynômes de $\mathbb{K}[X]$ de degrés $m = \deg(A)$ et $n = \deg(B)$.

1. Soit $A = QB + R$ la division euclidienne dans $\mathbb{K}[X]$ de A par B. Si $r = \deg(R)$, et si b_n est le coefficient dominant de B, montrer que
$$\mathrm{Res}(A,B) = (-1)^{mn} \cdot b_n^{m-r} \cdot \mathrm{Res}(B,R).$$
2. Écrire un algorithme de type Euclide (c'est-à-dire à base de divisions répétées), qui calcule le résultant Res(A, B) en $O(mn)$ opérations dans $\mathbb{K}$.

■

Calcul avec des nombres algébriques

L'idée que les polynômes sont de bonnes structures de données pour représenter leur racines amène à chercher des algorithmes pour effectuer les opérations de base sur ces racines, comme la somme ou le produit. Le résultant répond à cette attente.

Proposition 6.10 Soient $A = \prod_i (X - \alpha_i)$, $B = \prod_j (X - \beta_j)$ et C des polynômes unitaires de $\mathbb{K}[X]$, avec C et A premiers entre eux. Alors

$$\begin{aligned}
\mathrm{Res}_X(A(X), B(T - X)) &= \prod_{i,j} (T - (\alpha_i + \beta_j)),\\
\mathrm{Res}_X(A(X), B(T + X)) &= \prod_{i,j} (T - (\beta_j - \alpha_i)),\\
\mathrm{Res}_X(A(X), X^{\deg B} B(T/X)) &= \prod_{i,j} (T - \alpha_i \beta_j),\\
\mathrm{Res}_X(A(X), C(X)T - B(X)) &= \mathrm{Res}(A, C) \prod_i \left(T - \frac{B(\alpha_i)}{C(\alpha_i)}\right).
\end{aligned}$$

Démonstration. C'est une application directe du Théorème 6.5. ■

Cette proposition fournit une preuve effective du résultat suivant.

Corollaire 6.11 L'ensemble des éléments algébriques sur $\mathbb{K}$ est un corps.

Exemple 6.14 On sait que $\sqrt{2}$ est racine de $X^2 - 2$, tout comme $\sqrt{3}$ est racine de $X^2 - 3$. Un polynôme ayant pour racine $\sqrt{2} + \sqrt{3}$ est donné par

$$\mathrm{Res}_X(X^2 - 2, (T - X)^2 - 3) = T^4 - 10T^2 + 1.$$

Ces opérations ne distinguent pas les racines de A et B, les quatre racines du résultant sont donc $\pm\sqrt{2} \pm \sqrt{3}$.

Exemple 6.15 Ramanujan a prouvé l'identité suivante

$$\sqrt[3]{\cos\frac{2\pi}{7}} + \sqrt[3]{\cos\frac{4\pi}{7}} + \sqrt[3]{\cos\frac{8\pi}{7}} = \sqrt[3]{\frac{5 - 3\sqrt[3]{7}}{2}}.$$

Pour prouver automatiquement cette identité, une méthode consiste à construire un polynôme annulant son membre gauche, puis à isoler la racine correspondant numériquement à sa valeur, et à vérifier qu'il s'agit bien du membre droit.

En posant $x = \exp(2i\pi/7)$, les trois racines cubiques sont annulées par les polynômes

$$P_1 = X^3 - \frac{x + x^{-1}}{2}, \quad P_2 = X^3 - \frac{x^2 + x^{-2}}{2}, \quad P_3 = X^3 - \frac{x^4 + x^{-4}}{2}.$$

Un premier polynôme $P_4(Y) = \mathrm{Res}_X(P_1(Y - X), P_2(X))$ de degré 9 est d'abord obtenu, puis $P_5 = \mathrm{Res}_X(P_4(Y - X), P_3(X))$ de degré 27, et enfin le résultant de $x^7 - 1$ et du

numérateur de P_5 vu maintenant comme polynôme en x fournit

$$\begin{aligned}32768Y^6(Y-3)(Y^2+3)^2(Y^2+3Y+9)(Y^2+3Y+3)^3\\ \times(Y^2-3Y+3)^3(4Y^9-30Y^6+75Y^3+32)^6\\ \times(64Y^{18}+768Y^{15}+7728Y^{12}+17152Y^9+16008Y^6+3864Y^3+343)^6.\end{aligned}$$

Une évaluation numérique montre que c'est le facteur

$$4Y^9-30Y^6+75Y^3+32$$

qui annule le membre gauche (il faut faire un peu attention : Ramanujan utilise la convention que la racine cubique d'un réel négatif est un réel négatif). On peut alors résoudre ce polynôme par les formules de Cardan pour découvrir le membre droit : la seule racine réelle est

$$-\sqrt[3]{\frac{3\sqrt[3]{7}-5}{2}}.$$

On a donc non seulement prouvé l'identité, mais découvert automatiquement le membre droit.

Exercice 6.6 Montrer à l'aide de résultants les identités suivantes :

$$\frac{\sin\frac{2\pi}{7}}{\sin^2\frac{3\pi}{7}}-\frac{\sin\frac{\pi}{7}}{\sin^2\frac{2\pi}{7}}+\frac{\sin\frac{3\pi}{7}}{\sin^2\frac{\pi}{7}}=2\sqrt{7}\qquad\text{et}\qquad\frac{\sin^2\frac{3\pi}{7}}{\sin\frac{2\pi}{7}}-\frac{\sin^2\frac{2\pi}{7}}{\sin\frac{\pi}{7}}+\frac{\sin^2\frac{\pi}{7}}{\sin\frac{3\pi}{7}}=0.$$

■

Calcul du résultant en deux variables

On dispose à l'heure actuelle d'algorithmes rapides (de complexité quasi-optimale) pour le calcul du résultant avec une seule variable. Le meilleur algorithme connu actuellement pour le calcul du résultant en deux variables est une méthode d'évaluation-interpolation qui se ramène à une seule variable (voir l'Exercice 6.10). Cet algorithme n'est pas quasi-optimal. En revanche, pour les trois premières opérations de la Proposition 6.10, des algorithmes quasi-optimaux à base de multiplication rapide de séries existent, ils ont été présentés au Chapitre 3 (Section 3.3, page 68).

Sous-résultants

Le nombre d'opérations dans le corps des coefficients n'est pas une mesure suffisante de la complexité des calculs lorsque les opérations elles-mêmes ont une complexité variable. C'est le cas pour le calcul de pgcd dans $\mathbb{Q}[X]$ et dans $\mathbb{K}(Y)[X]$. Dans ces corps de coefficients, on constate empiriquement les phénomènes suivants :

— l'algorithme d'Euclide amène à faire des divisions, et introduit des dénominateurs au cours du calcul ;
— la taille des coefficients croît rapidement ;
— ces coefficients peuvent souvent se « simplifier ».

Exemple 6.16 L'exécution de l'algorithme d'Euclide sur les polynômes

$$A = 115X^5 + 7X^4 + 117X^3 + 30X^2 + 87X + 44, \quad B = 91X^4 + 155X^3 + 3X^2 + 143X + 115$$

produit les restes successifs suivants :

$$\frac{3601622}{8281}X^3 - \frac{1196501}{8281}X^2 + \frac{151912}{637}X + \frac{2340984}{8281},$$
$$\frac{189886027626841}{12971681030884}X^2 - \frac{57448278681703}{3242920257721}X - \frac{17501090665331}{3242920257721},$$
$$\frac{3748556212578804983806085060}{435414847094570987735 1001}X + \frac{14183336091512396932801 4892}{33493449776505460595 0077}.$$

En multipliant ces restes par des constantes bien choisies, on obtient les restes :

$$3601622X^3 - 1196501X^2 + 1974856X + 2340984,$$
$$22930325761X^2 - 27749440252X - 8453612204,$$
$$288979986761465X + 142143002707719.$$

Il est souhaitable d'éviter le calcul sur les rationnels (ou les fractions rationnelles) : leur addition requiert des calculs de multiplications et éventuellement de pgcd. Une première idée est d'éviter totalement les divisions en utilisant des *pseudo-restes*.

Définition 6.2 Si A et B sont des polynômes de $\mathbb{A}[X]$ et si b est le coefficient de tête de B, le *pseudo-reste* $\bar{R}$ de A et B est défini par

$$b^{\deg A - \deg B + 1} A = \bar{Q}B + \bar{R},$$

où $\bar{Q}, \bar{R} \in \mathbb{A}[X]$ avec $\deg \bar{R} < \deg \bar{Q}$.

Remplacer les calculs de restes de l'algorithme d'Euclide par des calculs de pseudo-restes évite d'introduire des dénominateurs. Cette idée seule n'est pas suffisante : les coefficients de ces pseudo-restes croissent trop.

Exemple 6.17 Sur le même exemple, la modification de l'algorithme d'Euclide produit la suite de pseudo-restes

$$3601622X^3 - 1196501X^2 + 1974856X + 2340984,$$
$$189886027626841X^2 - 229793114726812X - 70004362661324,$$
$$257057096083899261191107914552182660X$$
$$+126440823512296156588839626542149756.$$

Une possibilité pour éviter cette croissance est de diviser ces polynômes par le pgcd de leurs coefficients à chaque étape. C'est ce que nous avons fait dans l'Exemple 6.16 ci-dessus. On parle alors de suite de pseudo-restes *primitifs*. Cependant, ces calculs de

Entrée A et B dans $\mathbb{A}[X]$ avec $\deg B < \deg A$.
Sortie Le dernier sous-résultant non nul.
1. Initialiser $f = g = s = 1$.
2. Tant que $\deg B > 0$ faire :
   a. $d = \deg A - \deg B$.
   b. Calculer le pseudo reste R de A par B.
   c. Si les degrés de A et B sont impairs, faire $s = -s$.
   d. Faire A := B et mettre dans B le quotient de la division de R par fg^d.
   e. Copier le coefficient dominant de A dans f.
   f. Mettre à jour g en lui affectant le quotient de la division de f^d par g^{d-1}.
3. $d = \deg A$;
4. Renvoyer le quotient de la division de sB^d par g^{d-1}.

Algorithme 6.6 – Algorithme des sous-résultants.

pgcd de coefficients sont trop coûteux.

Le calcul des sous-résultants, donné dans l'Algorithme 6.6, est une modification de l'algorithme d'Euclide qui évite les divisions, tout en *prévoyant* des facteurs communs qui apparaissent dans les coefficients de la suite des pseudo-restes de sorte à limiter leur croissance. Ces pseudo-restes sont appelés des sous-résultants. Dans cet algorithme, on se contente de renvoyer le dernier sous-résultant non nul ; on conçoit aisément comment modifier l'algorithme pour obtenir *n'importe* quel sous-résultant (spécifié par des conditions de degré, par exemple).

Exemple 6.18 Toujours sur les mêmes polynômes, la suite des sous-résultants redonne les polynômes que nous avions déjà calculés par simplification :

$$3601622X^3 - 1196501X^2 + 1974856X + 2340984,$$
$$22930325761X^2 - 27749440252X - 8453612204,$$
$$288979986761465X + 142143002707719.$$

La suite des sous-résultants de cet exemple est donc primitive : les facteurs prédits par l'algorithme du sous-résultant ont suffi à éliminer tout facteur commun entre les coefficients des pseudo-restes.

Comme le résultant, les sous-résultants sont liés à la matrice de Sylvester et à l'algorithme d'Euclide. Une formule de Cramer sur une sous-matrice de la matrice de Sylvester donne le résultat suivant.

Proposition 6.12 Toutes les divisions effectuées au cours de l'algorithme des sous-résultants sont exactes.

La preuve de ce résultat est technique et omise ici.

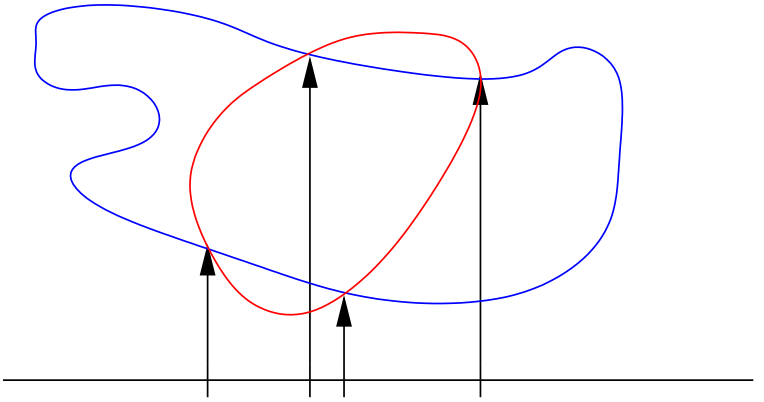


FIGURE 6.7 – Paramétrisation des solutions d'un système algébrique par les racines d'un résultant.

En corollaire, on remarque en outre qu'il est assez facile de déduire de ce résultat des estimations sur la « taille » des coefficients qui apparaissent au cours de l'algorithme. Un cas particulier intéressant est celui où $\mathbb{A}$ est l'anneau de polynômes $\mathbb{K}[Y]$. Alors, si A et B ont des degrés totaux m et n, tous les sous-résultants ont des degrés bornés par mn. Ces bornes permettent un calcul du résultant par évaluation-interpolation dès que l'on dispose d'un algorithme efficace à une variable.

Calcul de la paramétrisation

Les sous-résultants permettent de finir la « résolution » (commencée en page 118) des systèmes de deux polynômes à deux variables. Pour simplifier, nous faisons l'hypothèse que pour tout x racine du résultant $R = \mathrm{Res}_Y(A, B)$, il n'y a qu'un seul y tel que $A(x, y) = B(x, y) = 0$. Dans ce cas, on peut montrer que le sous-résultant S_1 (de degré 1 en Y) est non nul ; écrivons-le sous la forme $S_1 = P_0(X)Y + Q_0(X)$. Comme pour le résultant, il existe des polynômes U et V dans $\mathbb{K}[X, Y]$ tels qu'on ait l'égalité

$$AU + BV = P_0(X)Y + Q_0(X).$$

On en déduit donc que toutes les solutions (x, y) du système $A = B = 0$ satisfont l'équation $P_0(x)y = Q_0$. Autrement dit, en écartant les solutions dégénérées où $P_0(x) = 0$, on obtient l'ordonnée des points solutions en évaluant la fraction rationnelle $-Q_0/P_0$ sur les racines du résultant R. Autrement dit, cette procédure permet de décrire les solutions du système sous la forme

$$\begin{cases} y = Q(x) \\ R(x) = 0. \end{cases}$$

Géométriquement, ce polynôme Q permet de retrouver y à partir de x, il permet donc d'effectuer l'opération symbolisée sur la Figure 6.7, la paramétrisation des solutions du système par les racines de R.

Exemple 6.19 Sur les mêmes polynômes qu'à l'Exemple 6.8, le sous-résultant redonne la paramétrisation calculée à partir de l'algorithme d'Euclide, mais en étant plus économe dans les opérations sur les coefficients.

6.3 Algorithme d'Euclide rapide

Le point central de cette section est la présentation d'un algorithme rapide pour le pgcd, qui est un algorithme récursif à base de multiplications et de divisions euclidiennes rapides. Historiquement, c'est d'abord un algorithme rapide pour le pgcd d'entiers (le « demi-pgcd ») qui est apparu, il a ensuite été étendu aux polynômes. Nous décrivons maintenant l'algorithme dit du « demi-pgcd ». Pour simplifier, nous nous plaçons dans le cas du pgcd de polynômes, mais cette approche s'étend au cas des entiers.

Soient donc A et B dans $\mathbb{K}[X]$, avec $n = \deg A$ et $\deg B < n$ (ce qui n'est pas une forte restriction). Posons comme précédemment $R_0 = A$, $R_1 = B$. Soient ensuite R_i les restes successifs de l'algorithme d'Euclide et soit en particulier $R_N = \mathrm{pgcd}(A, B)$ le dernier non nul d'entre eux. D'après les égalités en page 115, il existe une matrice 2×2, notée $M_{A,B}$, de cofacteurs telle que :

$$M_{A,B}\begin{pmatrix} R_0 \\ R_1 \end{pmatrix} = \begin{pmatrix} U_N & V_N \\ U_{N+1} & V_{N+1} \end{pmatrix}\begin{pmatrix} R_0 \\ R_1 \end{pmatrix} = \begin{pmatrix} R_N \\ 0 \end{pmatrix}.$$

L'algorithme de pgcd rapide calcule d'abord cette matrice ; à partir de là, on en déduit aisément le pgcd, pour $O(M(n))$ opérations supplémentaires.

Notons qu'une étape de l'algorithme d'Euclide classique peut elle aussi s'écrire sous une forme matricielle. En effet, si Q_i est le quotient de la division de R_{i-1} par R_i, on peut écrire :

$$\begin{pmatrix} 0 & 1 \\ 1 & -Q_i \end{pmatrix}\begin{pmatrix} R_{i-1} \\ R_i \end{pmatrix} = \begin{pmatrix} R_i \\ R_{i+1} \end{pmatrix}.$$

Ainsi, la matrice $M_{A,B}$ est un produit de matrices élémentaires de ce type. Elle est inversible.

Demi-pgcd : définition

Comme étape intermédiaire pour obtenir une division euclidienne rapide, on utilise l'algorithme dit du « demi-pgcd » (*half-gcd* en anglais), qui permet de faire des « pas de géant » dans la liste des restes successifs.

Le demi-pgcd est défini comme suit. Les degrés des restes successifs R_i décroissent strictement, donc il existe un unique indice j tel que :

— $\deg R_j \geq \frac{n}{2}$;
— $\deg R_{j+1} < \frac{n}{2}$.

On a vu en page 115 qu'il existe $U_j, V_j, U_{j+1}, V_{j+1}$ tels que :

$$U_j R_0 + V_j R_1 = R_j \quad \text{et} \quad U_{j+1} R_0 + V_{j+1} R_1 = R_{j+1},$$

Entrée $R_0 = A$ et $R_1 = B$ dans $\mathbb{K}[X]$ avec $\deg B < \deg A = n$.
Sortie La matrice $M_{A,B}$.

1. Soit $M_{\text{dpgcd}} = \text{dpgcd}(A, B)$.
2. Calculer R_j et R_{j+1}.
3. Si $R_{j+1} = 0$, renvoyer M_{dpgcd}.
4. Soit M la matrice de la division euclidienne de R_j par R_{j+1}, et R_{j+2} le reste correspondant.
5. Si $R_{j+2} = 0$, renvoyer MM_{dpgcd}.
6. Calculer $M_{R_{j+1},R_{j+2}}$ par un appel récursif.
7. Renvoyer $M_{R_{j+1},R_{j+2}} MM_{\text{dpgcd}}$.

Algorithme 6.8 – Algorithme d'Euclide rapide via le demi-pgcd rapide.

ces polynômes étant en outre uniques si on rajoute les conditions de degré voulues sur la relation de Bézout. Ceci peut se récrire sous la forme matricielle suivante :

$$\begin{pmatrix} U_j & V_j \\ U_{j+1} & V_{j+1} \end{pmatrix} \begin{pmatrix} R_0 \\ R_1 \end{pmatrix} = \begin{pmatrix} R_j \\ R_{j+1} \end{pmatrix}.$$

Pour fixer les idées, en première approximation, on peut estimer que les polynômes $U_j, V_j, U_{j+1}, V_{j+1}$ ont des degrés de l'ordre de $\frac{n}{2}$ (attention, ce n'est qu'une estimation, pas une égalité).

Notons M_{dpgcd} la matrice ci-dessus donnant les restes R_j et R_{j+1}. L'algorithme du demi-pgcd (noté dpgcd ci-dessous) a pour vocation de calculer cette matrice. Avant d'en étudier le fonctionnement, voyons comment il permet d'obtenir le calcul du pgcd étendu. L'idée est qu'un appel à dpgcd en degré n permet d'obtenir les restes de degré approximativement $\frac{n}{2}$; alors, un appel en degré $\frac{n}{2}$ permet d'obtenir les restes de degré approximativement $\frac{n}{4}$, et ainsi de suite jusqu'à trouver le pgcd. Cette approche est détaillée dans l'Algorithme 6.8. À l'étape (4), R_j et R_{j+1} ont par définition des degrés qui encadrent $\frac{n}{2}$, mais on n'a pas de borne supérieure sur le degré de R_j, qui peut être proche de n. Aussi, pour obtenir deux polynômes de degré au plus $\frac{n}{2}$ pour l'appel récursif, il est nécessaire d'effectuer une étape de division euclidienne.

Soit $H(n)$ la complexité de l'algorithme de demi-pgcd sur des entrées de degré au plus n. Pour estimer la complexité de l'algorithme de pgcd rapide, on fait l'hypothèse que $H(n+n') \geq H(n) + H(n')$ et que $M(n)$ est négligeable devant $H(n)$. Ces hypothèses sont vérifiées pour l'algorithme proposé en Section 6.3.

Proposition 6.13 L'Algorithme 6.8 calcule $M_{A,B}$ en $O(H(n))$ opérations.

Démonstration. La validité de l'algorithme est immédiate, puisque toutes les matrices calculées ne sont au fond que des matrices qui font la transition de deux restes successifs à deux autres restes successifs. Multiplier ces matrices permet de composer ces opérations de transition.

Estimons la complexité. Le coût du dpgcd est $H(n)$. Ensuite, toutes les opérations des étapes (2) à (5) ont une complexité en $O(M(n))$, en utilisant pour l'étape (4) une

Entrée A et B dans $\mathbb{K}[X]$, avec $\deg B < \deg A = n$.
Sortie La matrice M_{dpgcd} de A et B.

1. Soit $m = \lceil \frac{n}{2} \rceil$. Si $\deg B < m$, renvoyer la matrice identité.
2. Soient f et g les quotients respectifs de A et B par X^m (*qui ont des degrés approximativement* $n/2$).
3. Soient $M = \text{dpgcd}(f, g)$, et A', B' définis par
$$M\begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} A' \\ B' \end{pmatrix}.$$
4. Si $\deg B' < m$, renvoyer M.
5. Soient $C' = A' \bmod B'$, et M' la matrice de la division euclidienne correspondante.
Informellement, B' *a un degré au pire de l'ordre de* $\frac{3n}{4}$. *Par contre,* A' *peut avoir un degré plus grand. L'étape de division euclidienne permet de se ramener à des polynômes de degrés de l'ordre de* $\frac{3n}{4}$.
6. Soient $\ell = 2m - \deg B'$, b et c les quotients respectifs de B' et C' par X^ℓ.
ℓ *est de l'ordre de* $\frac{n}{4}$, *de sorte que b et c ont des degrés de l'ordre de* $\frac{n}{2}$.
7. Soit $M'' = \text{dpgcd}(b, c)$. Renvoyer la matrice $M''M'M$.

Algorithme 6.9 – Algorithme du demi-pgcd (dpgcd) rapide.

division euclidienne rapide (cf. Théorème 4.1 en page 83). On effectue ensuite un appel récursif, puis de nouveau des opérations dont le coût est en $O(M(n))$. Ainsi, la complexité $B(n)$ de l'algorithme de pgcd rapide satisfait à la récurrence :

$$B(n) \leq B(n/2) + H(n) + CM(n),$$

C étant une constante. Le théorème « diviser pour régner » permet de conclure. ■

Autrement dit, le coût du pgcd est, à une constante près, le même que celui du demi-pgcd. Il devient donc légitime de se consacrer uniquement à ce dernier.

Demi-pgcd : algorithme

Pour mettre en place une stratégie « diviser pour régner » pour le demi-pgcd, on utilise un moyen de « couper les polynômes » en deux. L'idée est de ne garder que les coefficients de poids forts des polynômes d'entrée et d'utiliser le fait que *le quotient de la division euclidienne de deux polynômes ne dépend que de leurs coefficients de poids fort*, d'une façon tout à fait quantifiée par la suite.

Remarquons qu'on accède aux coefficients de poids fort d'un polynôme en prenant son quotient par un polynôme de la forme X^k : par exemple, si A est

$$X^{10} + 2X^9 + 5X^8 + 3X^7 + 11X^6 + \cdots,$$

le quotient de A par X^6 est

$$X^4 + 2X^3 + 5X^2 + 3X + 11.$$

Voyons comment cette idée permet d'obtenir notre algorithme. Étant donnés deux polynômes A et B de degrés de l'ordre de n, on leur associe f et g de degré approximativement $\frac{n}{2}$, en nommant f et g les quotients respectifs de A et B par $X^{\frac{n}{2}}$: on a jeté les coefficients de petits degrés.

On calcule la matrice M du demi-pgcd de f et g (grosso modo, les éléments de cette matrice ont degré $\frac{n}{4}$). La remarque ci-dessus permet de montrer qu'on obtient ainsi une matrice de transition vers les restes de A et B eux-mêmes.

On applique donc cette matrice aux polynômes initiaux ; on obtient des restes B′ et C′ dont les degrés sont de l'ordre de $\frac{3n}{4}$. On effectue alors une seconde fois le même type d'opération, avec un appel récursif en degré $\frac{n}{2}$, pour gagner à nouveau $\frac{n}{4}$, et finalement arriver en degré $\frac{n}{2}$.

Les choix exacts des degrés de troncature sont subtils, et on admettra que les valeurs données dans l'Algorithme 6.9 sont correctes.

Proposition 6.14 L'Algorithme 6.9 calcule la matrice du demi-pgcd de A et B en $O(M(n)\log n)$ opérations dans $\mathbb{K}$.

Démonstration. Comme indiqué plus haut, on admet que les choix des degrés de troncature permettent d'assurer la validité de l'algorithme. Il est plus facile d'en effectuer l'analyse de complexité. Le coût $H(n)$ peut être majoré par deux fois le coût en degré au plus $\frac{n}{2}$ (les deux appels ont lieu lors des étapes (3) et (7)), plus un certain nombre de multiplications de polynômes et une division euclidienne. En remarquant que tous les produits se font en degré au plus n, on en déduit la récurrence

$$H(n) \leq 2H(n/2) + CM(n),$$

où C est une constante. La conclusion découle comme d'habitude du lemme « diviser pour régner ». ■

Proposition 6.15 Le calcul du résultant de deux polynômes A et B à coefficients dans $\mathbb{K}$ peut se faire en $O(M(n)\log n)$ opérations dans $\mathbb{K}$.

Démonstration. Il s'agit d'adapter l'Algorithme 6.9 pour accélérer l'algorithme de coût quadatrique vu dans l'Exercice 6.5. Nous laissons ces détails techniques en exercice. ■

Complément : calcul d'un reste choisi

On peut tirer un raffinement fort utile de l'algorithme de demi-pgcd : le calcul d'un reste particulier (sélectionné par une condition de degré), ainsi que des cofacteurs associés.

Théorème 6.16 Soient $R_0 = A$ et $R_1 = B$, avec $\deg A = n$ et $\deg B < \deg A$, et soient R_i les restes successifs de l'algorithme d'Euclide appliqué à A et B.

Soit $\ell < n$, et R_j le premier reste de degré inférieur à ℓ. On peut calculer R_j et les cofacteurs associés U_j et V_j en $O(M(n)\log n)$ opérations de $\mathbb{K}$.

Démonstration succinte. Si ℓ est plus petit que $\frac{n}{2}$, on fait un calcul de dpgcd pour se ramener en degré $\frac{n}{2}$, et on effectue un appel récursif. Si ℓ est plus grand que $\frac{n}{2}$, on fait un calcul de dpgcd sur les polynômes divisés par $X^{n-2\ell}$. ■

Le cas des entiers

Les mêmes idées algorithmiques s'étendent au calcul sur les entiers, mais il est beaucoup plus difficile d'écrire un algorithme correct dans ce cas. On se contentera d'énoncer les résultats de complexité suivants :

Théorème 6.17 Soient $R_0 = A \in \mathbb{N}$, $R_1 = B \in \mathbb{N}$, avec $B \leq A$ et R_i les restes de l'algorithme d'Euclide appliqué à A et B. On peut calculer :

— le pgcd de A et B,

— le premier reste inférieur à un entier $\ell < A$,

ainsi que les cofacteurs associés en $O(M_{\mathbb{Z}}(\log A)\log\log A)$ opérations binaires.

Exercices

Exercice 6.7 — Factorisation sans carré. Soit $\mathbb{K}$ un corps de caractéristique nulle. Si $F \in \mathbb{K}[X]$ se factorise en irréductibles comme $\prod_i f_i^{\alpha_i}$, le polynôme $\bar{F} = \prod_i f_i$ est appelé la *partie sans carré de* F. Si $n = \deg(F)$, montrer qu'on peut calculer les coefficients de la partie sans carré de F à partir des coefficients de F en $O(M(n)\log n)$ opérations dans $\mathbb{K}$. ■

Exercice 6.8 Soient $f, g \in \mathbb{K}[X]$ des polynômes unitaires.

1. Soit $N \in \mathbb{N}\setminus\{0\}$. Montrer que l'unique polynôme unitaire de $\mathbb{K}[X]$ dont les racines sont les puissances N-ièmes des racines de f peut être obtenu par un calcul de résultant.
2. Si f est le polynôme minimal d'un nombre algébrique α, montrer qu'on peut déterminer un polynôme annulateur de $g(\alpha)$ à l'aide d'un résultant.
3. Calculer le polynôme minimal sur $\mathbb{Q}$ de $\alpha = \sqrt[3]{4} + \sqrt[3]{2} + 1$.

■

Exercice 6.9 Soit $f \in \mathbb{K}[X]$ un polynôme unitaire de degré $d \geq 1$. Pour $N \geq 1$, on note $G_N(f)$ l'unique polynôme unitaire de degré d dont les racines sont les puissances N-ièmes des racines de f.

1. Exprimer $G_N(f)$ à l'aide d'un résultant de polynômes à deux variables.
2. Justifier l'appartenance à $\mathbb{K}[X]$ du polynôme $G_N(f)$.
3. Utiliser la question (1) pour donner un algorithme pour le calcul de $G_N(f)$;

estimer sa complexité.
4. Montrer que $G_2(f)$ peut se calculer en $O(M(d))$ opérations dans $\mathbb{K}$.
5. Si N est une puissance entière de 2, montrer comment on peut calculer $G_N(f)$ en $O(M(d)\log N)$ opérations dans $\mathbb{K}$.

■

Exercice 6.10 — Résultants en deux variables. Soient $F, G \in \mathbb{K}[X, Y]$ ayant des degrés en X et en Y bornés par un entier $D \geq 1$. Montrer que le polynôme $R(Y) := \mathrm{Res}_X(F, G)$ est de degré au plus $2D^2$. Donner un algorithme de type évaluation-interpolation pour le calcul des coefficients de R et estimer son coût en opérations dans $\mathbb{K}$. ■

Exercice 6.11 — Résultant de Rothstein–Trager. Soit $F = A/B$ une fraction rationnelle dans $\mathbb{Q}(X)$, où A et B sont deux polynômes de $\mathbb{Q}[X]$, premiers entre eux et tels que $\deg(A) < \deg(B)$. On suppose que tous les pôles $p_1, \ldots, p_\ell$ de F sont simples. La décomposition en éléments simples de F s'écrit $F = \sum_{i=1}^{\ell} r_i/(X - p_i)$, où r_i est le *résidu* de F au pôle p_i.
1. Montrer que pour tout i, le résidu r_i vaut $A(p_i)/B'(p_i)$.
2. Expliciter, à l'aide d'un résultant, un polynôme R qui annule tous les résidus $r_1, \ldots, r_\ell$.
3. Donner un algorithme permettant le calcul des coefficients de R à partir de ceux de A et de B, et estimer sa complexité arithmétique en fonction du degré de B.

■

Notes

Pour les déterminants sur des anneaux, nous renvoyons au livre de Lang [Lan02]. L'introduction du résultant et aussi des sous-résultants remonte à Bézout et Sylvester [Béz64 ; Syl39 ; Syl40]. Ces notions ont été introduites en calcul formel dans les années 1965 par Collins [Col66 ; Col67 ; Col71], Brown et Traub [Bro71 ; BT71]. L'article de synthèse de von zur Gathen et Lücking retrace leur histoire [GL03]. De nombreuses propriétés des résultants et de l'algorithme d'Euclide sont établies de manière élémentaire à l'aide des fonctions symétriques par Lascoux [Las03]. L'article d'El Kahoui est aussi une référence très abordable pour l'algorithme des sous-résultants [El 03]. On peut aussi consulter les livres suivants pour une approche un peu plus complète : [GCL92 ; GG03 ; Knu97 ; Lan02].

L'idée de base l'algorithme de demi-pgcd (le quotient de la division euclidienne de deux entiers ne dépend que de leurs chiffres de poids fort) est due à Lehmer [Leh38]. Knuth [Knu71] en a fait le premier algorithme sous-quadratique, qui calcule le pgcd de deux entiers de taille binaire n en complexité binaire $O(M_{\mathbb{Z}}(n)\log^4 n)$. Schönhage [Sch71] a montré comment abaisser cette complexité à $O(M_{\mathbb{Z}}(n)\log n)$. Son algorithme a été adapté au cas polynomial par Moenck [Moe73], mais seulement pour le cas où la suite des restes est normale (de degré décroissant régulièrement de 1). Brent, Gustavson et Yun [BGY80 ; GY79] ont donné un algorithme de type LKS (Lehmer–Knuth–Schönhage) pour le cas polynomial général, de complexité arithmétique $O(M(n)\log n)$ en degré n. Cet algorithme utilise $O(n \log n)$ opérations non scalaires. Strassen [Str81 ; Str83] a montré que $\Omega(n \log n)$ opérations non scalaires sont

aussi nécessaires, du moins pour une entrée générique, donc l'algorithme LKS est optimal de ce point de vue.

La proposition 6.15 est due à Schwartz [Sch80b]. Schönhage a donné un algorithme (probabiliste) pour le calcul rapide du pgcd de deux polynômes de $\mathbb{Z}[X]$. Pour deux polynômes de degré borné par n et coefficients d'au plus ℓ chiffres, l'algorithme a une complexité binaire $\tilde{O}(n(n+\ell))$, il est donc quasi-optimal (en la taille binaire de l'entrée) lorsque $\ell > n$ [Sch88b].

L'Exercice 6.7 est inspiré par un résultat de Yun [Yun76], qui montre qu'il est possible de déterminer toute la factorisation sans carré en la même complexité. En fait, les problèmes du calcul du pgcd et de la factorisation sans carré sont équivalents du point de vue de la complexité [Yun77]. Ces résultats sont abordés dans le Chapitre 18.

Il est très difficile de trouver une référence complète, lisible et sans erreur sur les algorithmes de pgcd rapide. Le Chapitre 11 de la troisième édition de l'ouvrage de von zur Gathen et Gerhard [GG13] fournit une bonne partie des résultats techniques implicitement utilisés dans la Section 6.3. Les notes du livre de Yap [Yap00] nous ont beaucoup inspirés, mais elles contiennent des erreurs pour le pgcd entier. On pourra également consulter les articles plus récents suivants : [Möl08 ; PW02 ; SZ04].

Ces algorithmes de pgcd rapide sont aussi assez délicats à implanter de manière efficace. Par exemple, pour le pgcd de polynômes, une bonne implantation doit prendre en compte les algorithmes de multiplication utilisés. Si on utilise la FFT, il est possible d'économiser des transformées de Fourier directes et inverses ; il faut alors régler le nombre de points d'évaluation en fonction des degrés du résultat final, et pas des résultats intermédiaires, etc. Si l'on travaille sur un corps fini « raisonnable » (les coefficients faisant de quelques bits jusqu'à quelques dizaines voire centaines de bits), on peut estimer que le seuil au-delà duquel utiliser l'algorithme rapide se situe autour des degrés 200 ou 300. Il faut noter que très peu de systèmes disposent de telles implantations (c'est le cas pour Magma, Mathemagix, et la bibliothèque NTL). En ce qui concerne le pgcd des entiers, la situation est sensiblement plus délicate, toujours à cause du problème des retenues (une bonne partie des algorithmes présentés dans les ouvrages de référence sont incorrects, à cause de ce problème). Pour donner une idée, dans les systèmes Magma, Mathematica, ou des bibliothèques telles que GMP (code toujours en développement et utilisé entre autres par les entiers de Maple, Mathematica et Sage), le seuil se situe autour de nombres de 30 000 bits (10 000 chiffres décimaux). Les notions de ce chapitre — pgcd, pgcd étendu, demi-pgcd, résultant, sous-résultants — peuvent aussi être définies dans le contexte non commutatif des récurrences et équations différentielles. Certaines de ces généralisations seront présentées dans le cas linéaire au Chapitre 30 et on trouvera plus de notes en page 558.

Bibliographie

Béz64 Bézout, É. (1764). « Recherches sur le degré des équations résultantes de l'évanouissement des inconnues ». In : *Histoire de l'académie royale des sciences*, p. 288–338.

BGY80 Brent, Richard P., Fred G. Gustavson et David Y. Y. Yun (1980). « Fast solution of Toeplitz systems of equations and computation of Padé approximants ». In : *Journal of Algorithms*, vol. 1, n°3, p. 259–295.

Bro71 Brown, W. S. (1971). « On Euclid's algorithm and the computation of polynomial greatest common divisors ». In : *SYMSAC'71 : ACM Symposium on Symbolic and Algebraic Manipulation*. Los Angeles, California, United States : ACM, p. 195–211.

BT71 Brown, W. S. et J. F. Traub (1971). « On Euclid's algorithm and the theory of subresultants ». In : *Journal of the Association for Computing Machinery*, vol. 18, p. 505–514.

Col66 Collins, G. E. (1966). « Polynomial remainder sequences and determinants ». In : *The American Mathematical Monthly*, vol. 73, p. 708–712.

Col67 Collins, George E. (1967). « Subresultants and reduced polynomial remainder sequences ». In : *Journal of the Association for Computing Machinery*, vol. 14, p. 128–142.

Col71 — (1971). « The calculation of multivariate polynomial resultants ». In : *Journal of the Association for Computing Machinery*, vol. 18, p. 515–532.

El 03 El Kahoui, M'hammed (2003). « An elementary approach to subresultants theory ». In : *Journal of Symbolic Computation*, vol. 35, n°3, p. 281–292.

GCL92 Geddes, Keith O., Stephen R. Czapor et George Labahn (1992). *Algorithms for computer algebra*. Kluwer Academic Publishers.

GG03 Gathen, Joachim von zur et Jürgen Gerhard (2003). *Modern computer algebra*. 2e éd. Cambridge University Press.

GG13 — (2013). *Modern computer algebra*. 3e éd. Cambridge University Press.

GL03 Gathen, Joachim von zur et Thomas Lücking (2003). « Subresultants revisited ». In : *Theoretical Computer Science*, vol. 297, n°1-3, p. 199–239.

GY79 Gustavson, Fred G. et David Y. Y. Yun (1979). « Fast algorithms for rational Hermite approximation and solution of Toeplitz systems ». In : *IEEE Transactions on Circuits and Systems*, vol. 26, n°9, p. 750–755.

Knu71 Knuth, Donald E. (1971). « The analysis of algorithms ». In : *Actes du Congrès International des Mathématiciens 1970*. Vol. 3. Nice : Gauthier-Villars, p. 269–274.

Knu97 — (1997). *The art of computer programming*. 3e éd. Vol. 2 : Seminumerical Algorithms. Computer Science and Information Processing. Addison-Wesley Publishing Co.

Lan02 Lang, Serge (2002). *Algebra*. 3e éd. Vol. 211. Graduate Texts in Mathematics. Springer-Verlag.

Las03 Lascoux, Alain (2003). *Symmetric functions and combinatorial operators on polynomials*. Vol. 99. CBMS Regional Conference Series in Mathematics. Published for the Conference Board of the Mathematical Sciences, Washington, DC.

Leh38 Lehmer, D. H. (1938). « Euclid's Algorithm for large numbers ». In : *The American Mathematical Monthly*, vol. 45, n°4, p. 227–233.

Moe73 Moenck, R. T. (1973). « Fast computation of GCDs ». In : *STOC'73 : ACM Symposium on Theory of Computing*. Austin : ACM, p. 142–151.

Möl08 Möller, Niels (2008). « On Schönhage's algorithm and subquadratic integer GCD computation ». In : *Mathematics of Computation*, vol. 77, n°261, p. 589–607.

PW02 Pan, V. Y. et X. Wang (2002). « Acceleration of Euclidean algorithm and extensions ». In : *ISSAC'02 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Teo Mora. Lille : ACM Press, p. 207–213.

Sch71 Schönhage, A. (1971). « Schnelle Berechnung von Kettenbruchentwicklugen ». In : *Acta Informatica*, vol. 1, p. 139–144.

Sch80b Schwartz, J. T. (1980). « Fast probabilistic algorithms for verification of polynomial identities ». In : *Journal of the Association for Computing Machinery*, vol. 27, n°4, p. 701–717.

Sch88b Schönhage, A. (1988). « Probabilistic computation of integer polynomial GCDs ». In : *Journal of Algorithms*, vol. 9, n°3, p. 365–371.

Str81 Strassen, V. (1981). « The computational complexity of continued fractions ». In : *SYMSAC'81*. Snowbird, Utah, United States : ACM, p. 51–67.

Str83 — (1983). « The computational complexity of continued fractions ». In : *SIAM Journal on Computing*, vol. 12, n°1, p. 1–27.

Syl39 Sylvester, James Joseph (1839). « On rational derivation from equations of coexistence, that is to say, a new and extended theory of elimination, *1839–40* ». In : *The Collected mathematical papers of James Joseph Sylvester*. Baker, H. F., p. 40–53.

Syl40 — (1840). « A method of determining by mere inspection the derivatives from two equations of any degree ». In : *Philosophical Magazine Series 3*, vol. 16, n°101, p. 132–135.

SZ04 Stehlé, D. et P. Zimmermann (2004). « A binary recursive GCD algorithm ». In : *ANTS-VI*. Vol. 3076. Lecture Notes in Computer Science. Springer-Verlag, p. 411–425.

Yap00 Yap, Chee (2000). *Fundamental problems in algorithmic algebra*. Oxford University Press.

Yun76 Yun, David Y. Y. (1976). « On square-free decomposition algorithms ». In : *SYMSAC'76 : ACM Symposium on Symbolic and Algebraic Computation*. Yorktown Heights, New York, United States : ACM, p. 26–35.

Yun77 — (1977). « On the equivalence of polynomial GCD and square-free factorization problems ». In : *MACSYMA Users' Conference*. NASA, Langley Res. Center, Washington D.C., United States, p. 65–70.

7. Approximants de Padé et de Padé–Hermite

Résumé

Un premier problème abordé dans ce chapitre est le calcul d'approximants de Padé. Plus généralement, on s'intéresse à la *reconstruction rationnelle*, dont un autre cas particulier important est l'interpolation des fractions rationnelles. En Section 7.1, nous ramenons la reconstruction rationnelle à l'algorithme d'Euclide étendu, et l'appliquons à la reconnaissance d'une suite récurrente linéaire à partir de ses premiers termes (algorithme de Berlekamp–Massey.) Les applications en sont nombreuses : pour la résolution de systèmes différentiels à coefficients constants (Chapitre 13), pour le calcul de polynômes minimaux de matrices creuses (Chapitre 9), pour la résolution de systèmes linéaires à coefficients polynomiaux (Chapitre 11).

Un second problème, traité en Section 7.2, est le calcul d'approximants de Padé–Hermite. Il peut s'effectuer grâce à un algorithme qui généralise le calcul des approximants de Padé. L'approximation de Padé–Hermite permet de reconstruire des polynômes annulant une série algébrique, et des opérateurs différentiels annulant une série différentiellement finie. Il permet également de reconstruire une récurrence à coefficients polynomiaux d'ordre arbitraire à partir des premiers termes de la suite.

7.1 Reconstruction rationnelle

Définition 7.1 Soit $\mathbb{K}$ un corps, $A \in \mathbb{K}[X]$ un polynôme de degré $n > 0$ et $B \in \mathbb{K}[X]$ de degré $< n$. Pour un $k \in \{1, \ldots, n\}$ fixé, la *reconstruction rationnelle de* B *modulo* A

est la recherche d'un couple de polynômes $(R, V) \in \mathbb{K}[X]^2$ vérifiant :

$$\text{pgcd}(V, A) = 1, \quad \deg(R) < k, \quad \deg(V) \leq n-k \quad \text{et} \quad \frac{R}{V} \equiv B \bmod A. \qquad \text{(RR)}$$

On parle d'*approximation de Padé* lorsque $A = X^n$, et d'*interpolation de Cauchy* lorsque $A = \prod_i (X - u_i)$ avec $u_1, \ldots, u_n \in \mathbb{K}$ deux à deux distincts.

R Si $k = n$, alors clairement $(R, V) = (B, 1)$ est une solution du problème (RR).
Si $k < n$, il est possible que (RR) n'admette aucune solution. C'est le cas par exemple en prenant $n = 3, k = 2$ et $A = X^3, B = X^2 + 1 \in \mathbb{K}[X]$. En effet, si $V(X) = aX + b$ avec $b \neq 0$, alors $R \equiv (aX + b)(X^2 + 1) = bX^2 + aX + b \bmod X^3$, ce qui est incompatible avec $\deg(R) \leq 1$.
Par ailleurs, si (RR) admet une solution (R, V), alors la fraction rationnelle R/V est forcément unique. En effet, si $(R_1, V_1) \in \mathbb{K}[X]^2$ est une autre solution de (RR), alors $R_1/V_1 \equiv R/V \bmod A$, donc A divise $R_1 V - V_1 R$. Or, le polynôme $R_1 V - V_1 R$ ayant un degré strictement inférieur à celui de A, il doit être identiquement nul. Donc les fractions R/V et R_1/V_1 coïncident.

Un problème plus simple

Si $R, V \in \mathbb{K}[X]$ sont tels que (R, V) est solution du problème (RR), alors (R, V) vérifie aussi le problème plus simple

$$\deg(R) < k, \quad \deg(V) \leq n-k \quad \text{et} \quad R \equiv VB \bmod A, \qquad \text{(RRS)}$$

où, à la différence de (RR), on a mis de côté la contrainte sur le pgcd de A et de V.

Remarquons tout de suite que le problème (RRS) admet *toujours* une solution non triviale $(R, V) \neq (0, 0)$: en effet, il se traduit en termes d'algèbre linéaire en un système linéaire homogène ayant $k + (n - k + 1) = n + 1$ inconnues (les coefficients de R et de V) et n équations. Par ailleurs, la solution R/V du problème (RRS) est encore unique (même idée de preuve que pour (RR) : si A divise à la fois $R - VB$ et $R_1 - V_1 B$, alors il divise également la combinaison linéaire $RV_1 - R_1 V = (R - VB)V_1 - (R_1 - V_1 B)V$).

Lien avec le pgcd étendu

Nous allons prouver que le problème (RRS) peut être résolu en utilisant l'algorithme d'Euclide étendu décrit en page 116. Nous en déduirons ensuite une procédure de décision et calcul pour (RR).

L'intuition du lien entre le problème (RRS) et le calcul du pgcd étendu s'appuie sur la remarque simple suivante : la congruence $R \equiv VB \bmod A$ équivaut à l'existence d'un polynôme U tel que $R = UA + VB$; or, cette dernière égalité est une *relation de type Bézout*.

Par exemple, dans le cas particulier $k = 1$, si A et B sont premiers entre eux, alors on prend $R = 1$ et la relation de Bézout $R = UA + VB$ avec $\deg(V) < \deg(A) = n$ fournit la réponse. Sinon, on prend $R = 0$, et comme le ppcm de A et B a un degré $< m + n$, le polynôme $V = \text{ppcm}(A, B)/B$ est de degré au plus $n - 1$ et vérifie $VB = 0 \bmod A$.

Calcul de la reconstruction rationnelle

L'algorithme d'Euclide étendu, décrit page 116, calcule une suite de restes successifs dont le degré décroît, ainsi qu'une suite de cofacteurs (U_i, V_i). Les éléments de la i-ième itération vérifient l'identité de Bézout $U_i A + V_i B = R_i$. L'écriture matricielle de l'itération

$$\begin{pmatrix} U_i & V_i \\ U_{i+1} & V_{i+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -Q_i \end{pmatrix} \times \begin{pmatrix} U_{i-1} & V_{i-1} \\ U_i & V_i \end{pmatrix}$$

permet de déduire aisément que $U_i V_{i+1} - V_i U_{i+1} = (-1)^i$ quel que soit i, et en particulier que U_i et V_i sont premiers entre eux. Cela entraîne également que $\gcd(R_i, V_i) = \gcd(A, V_i)$.

La clé du résultat principal de cette section (Théorème 7.2 ci-dessous) est le lemme suivant, qui montre que les cofacteurs V_i obtenus par l'algorithme d'Euclide étendu ont des degrés modérés et bien maîtrisés.

Lemme 7.1 Soient $n_0 = \deg(A) > n_1 = \deg(B) > n_2 > \ldots > n_\ell$ les degrés des restes R_i dans l'algorithme d'Euclide étendu exécuté avec A et B. Alors :

$$\deg(V_i) = n - n_{i-1} \quad \text{pour} \quad 1 \le i \le \ell + 1. \tag{7.1}$$

Démonstration. On procède par récurrence forte sur i. L'initialisation est claire, car $\deg(V_1) = 0$. Supposons l'égalité (7.1) vérifiée pour $1 \le i \le k$.

Comme $\deg(V_{k-1}) = n - n_{k-2} < n - n_{k-1} = \deg(V_k)$, le degré de $V_{k+1} = V_{k-1} - Q_k V_k$ est égal à $\deg(Q_k V_k) = \deg(Q_k) + \deg(V_k) = (n_{k-1} - n_k) + (n - n_{k-1}) = n - n_k$. Ceci donne le résultat pour $i = k + 1$ et conclut la preuve. ■

R Dans une situation « générique », les quotients successifs Q_i ont tous degré 1, et donc $n_{i+1} = n_i - 1$ (la suite des restes est appelée *normale*), si bien que le lemme précédent implique alors l'égalité $\deg(V_i) = i - 1$.

Le résultat suivant montre que problème de reconstruction rationnelle (RR) peut être résolu à l'aide de l'algorithme d'Euclide étendu.

Théorème 7.2 Soit $A \in \mathbb{K}[X]$ de degré $n > 0$ et soit $B \in \mathbb{K}[X]$ de degré $< n$. Soit $k \in \{1, 2, \ldots, n\}$ et soient R_j, U_j, V_j le reste et les cofacteurs obtenus à la j-ième itération de l'algorithme d'Euclide étendu exécuté sur A et B. Si j est choisi minimal tel que $\deg(R_j) < k$, alors on a :

1. Il existe une solution $(R, V) \neq (0, 0)$ de (RRS), à savoir $(R, V) = (R_j, V_j)$. Si, de plus, $\gcd(R_j, V_j) = 1$, alors (R, V) est aussi solution de (RR).
2. Si (RR) admet une solution et si $R/V \in \mathbb{K}(X)$ en est une forme irréductible, alors il existe une constante $\alpha \in \mathbb{K} \setminus \{0\}$ telle que $R = \alpha R_j$ et $V = \alpha V_j$.

Le problème (RR) admet donc une solution si, et seulement si, $\gcd(A, V_j) = 1$.

Démonstration. Par construction, R_j vaut bien $U_j A + V_j B \equiv V_j B$ modulo A. Par ailleurs, le Lemme 7.1 montre que le degré de V_j vaut $n - \deg(R_{j-1})$ et est donc borné par $n - k$ (par la minimalité de j). Donc $(R, V) = (R_j, V_j)$ vérifie bien (RRS).

Entrée $A, B \in \mathbb{K}[X]$ avec $n = \deg(A) > \deg(B)$, et $k \in \{1, \ldots, n\}$.
Sortie Une solution (R, V) de (RR), ou 0 si une telle solution n'existe pas.

1. $R_0 = A$; $U_0 = 1$; $V_0 = 0$; $R_1 = B$; $U_1 = 0$; $V_1 = 1$; $i = 1$.
2. Tant que $\deg(R_i) \geq k$, faire :
   a. Calculer la quotient Q_i et le reste R_{i+1} de la division de R_{i-1} par R_i.
   b. $U_{i+1} = U_{i-1} - Q_i U_i$; $V_{i+1} = V_{i-1} - Q_i V_i$.
   c. $i = i + 1$.
3. Si $\operatorname{pgcd}(A, V_i) = 1$, renvoyer (R_i, V_i) ; sinon renvoyer 0.

Algorithme 7.1 – Algorithme de reconstruction rationnelle.

De plus, le $\operatorname{pgcd}(A, V_j)$ est égal au $\operatorname{pgcd}(R_j, V_j)$. Par conséquent, si ce dernier vaut 1, alors V_j est inversible modulo A et donc $(R, V) = (R_j, V_j)$ vérifie aussi (RR).

L'unicité (à une constante près) est plus délicate à prouver. Supposons que (R, V) est une solution de (RR). Alors R s'écrit $UA + VB$ pour un certain $U \in \mathbb{K}[X]$. Nous allons prouver que $(R, V) = (\alpha R_j, \alpha V_j)$ pour un certain $\alpha \in \mathbb{K}[X] \setminus \{0\}$.

D'abord, les polynômes U_j et V_j vérifient $U_j V = U V_j$: supposons le contraire et considérons le système linéaire

$$\begin{pmatrix} U_j & V_j \\ U & V \end{pmatrix} \times \begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} R_j \\ R \end{pmatrix}.$$

Le système étant supposé de Cramer, A s'écrit comme quotient de deux déterminants

$$A = \begin{vmatrix} R_j & V_j \\ R & V \end{vmatrix} \Big/ \begin{vmatrix} U_j & V_j \\ U & V \end{vmatrix}.$$

L'hypothèse $\deg(R_j) < k \leq \deg(R_{j-1})$ et le Lemme 7.1 entraînent que le degré du membre droit de l'égalité précédente est majoré par

$$\begin{aligned} \deg(R_j V - R V_j) &\leq \max\{k - 1 + \deg(V), \deg(R) + n - \deg(R_{j-1})\} \\ &\leq \max\{n - 1, (k - 1) + n - \deg(R_{j-1})\} = n - 1, \end{aligned}$$

ce qui contredit $\deg(A) = n$, concluant la preuve de $U_j V = U V_j$. Cette égalité implique que V_j divise $U_j V$, et puisque U_j et V_j sont premiers entre eux, il s'ensuit que V_j divise V. En écrivant $V = \alpha V_j$, avec $\alpha \in \mathbb{K}[X]$, on obtient $U V_j = V U_j = \alpha U_j V_j$, et donc $U = \alpha U_j$. Enfin, $R = UA + VB = \alpha(U_j A + V_j B) = \alpha R_j$. Les polynômes R et V étant premiers entre eux, il s'ensuit que α est une constante de $\mathbb{K} \setminus \{0\}$, et que donc $\operatorname{pgcd}(R_j, V_j) = 1$. ■

L'Algorithme 7.1 résume cette approche ; la ressemblance avec l'algorithme d'Euclide étendu est claire.

Entrée Une borne $N \in \mathbb{N}$ sur le degré du polynôme minimal de la suite $(a_n)_{n\geq 0}$ et les 2N premiers termes $a_0, \dots, a_{2N-1} \in \mathbb{K}$.
Sortie Le polynôme générateur minimal de $(a_n)_{n\geq 0}$.
1. $A = a_0 + a_1 X + \cdots + a_{2N-1} X^{2N-1}$.
2. Calculer $R, V \in \mathbb{K}[X]$ la solution de (7.3) telle que $V(0) = 1$.
3. $d = \max\{1 + \deg(R), \deg(V)\}$. Renvoyer $\bar{V} = V(1/X)X^d$.

Algorithme 7.2 – Algorithme de Berlekamp–Massey.

Approximants de Padé

Si $n > 0$, $k \in \{1, 2, \dots, n\}$ et si $B \in \mathbb{K}[X]$ est de degré $< n$, alors un *approximant de Padé pour* B *de type* $(k-1, n-k)$ est une fraction rationnelle $R/V \in \mathbb{K}[X]$ telle que

$$X \nmid V, \quad \deg(R) < k, \quad \deg(V) \leq n-k \quad \text{et} \quad \frac{R}{V} \equiv B \bmod X^n. \tag{7.2}$$

Le Théorème 7.2 (pour $A = X^n$) entraîne la conséquence suivante.

Corollaire 7.3 Soit $B \in \mathbb{K}[X]$ de degré $< n$. Soient R_j, U_j, V_j le reste et les cofacteurs obtenus à la j-ième itération de l'algorithme d'Euclide étendu exécuté avec X^n et B. Si j est choisi minimal tel que $\deg(R_j) < k$, alors on a :
1. L'équation (7.2) admet une solution si et seulement si $\operatorname{pgcd}(R_j, V_j) = 1$.
2. Si $\operatorname{pgcd}(R_j, V_j) = 1$, alors R_j/V_j est l'unique approximant de Padé pour B de type $(k-1, n-k)$.

L'algorithme qui s'en déduit est un cas particulier de l'Algorithme 7.1, en prenant $A = X^n$ et en remplaçant le test $\operatorname{pgcd}(A, V_i) = 1$ de l'étape (3) par $V_i(0) \neq 0$.

Algorithme de Berlekamp–Massey

L'algorithme décrit dans cette section permet de deviner des récurrences à coefficients constants d'ordre arbitraire à partir de termes consécutifs de la suite.

On se donne dans un corps $\mathbb{K}$ les 2N premiers éléments d'une suite récurrente linéaire $(a_n)_{n\geq 0}$ pour laquelle on sait qu'il existe un polynôme caractéristique de degré $d \leq N$, c'est-à-dire un polynôme

$$f(X) = f_d X^d + \cdots + f_0, \quad \text{tel que} \quad f_d a_{n+d} + \cdots + f_0 a_n = 0, \quad \text{pour tout} \quad n \geq 0.$$

Le problème est de calculer le polynôme minimal (c'est-à-dire le polynôme caractéristique de degré minimal) de $(a_n)_{n\geq 0}$. Le Lemme 4.7 (page 87) montre que ce calcul équivaut à la résolution du problème de Padé

$$\frac{R}{V} \equiv A \bmod X^{2N}, \quad X \nmid V, \ \deg(R) < N, \ \deg(V) \leq N \ \text{ et } \ \operatorname{pgcd}(R, V) = 1. \tag{7.3}$$

En effet, il implique que $(R, V) = (N_0, \bar{P})$ est solution de (7.3). La méthode qui en découle est résumée dans l'Algorithme 7.2.

Interpolation rationnelle de Cauchy

L'interpolation des fractions rationnelles, appelée aussi *interpolation de Cauchy*, est une généralisation naturelle de l'interpolation polynomiale (de Lagrange).

Soient $k \in \{1,\dots,n\}$, $v_0,\dots,v_{n-1} \in \mathbb{K}$ et soient $u_0,\dots,u_{n-1}$ des points distincts de $\mathbb{K}$. On cherche une fraction rationnelle $R/V \in \mathbb{K}(X)$ avec $R, V \in \mathbb{K}[X]$ satisfaisant aux contraintes

$$V(u_i) \neq 0, \quad \frac{R(u_i)}{V(u_i)} = v_i \quad \text{pour } 0 \leq i < n, \quad \deg(R) < k,\ \deg(V) \leq n-k. \tag{7.4}$$

Le Théorème 7.2 implique la procédure de décision et de calcul suivante : on calcule le polynôme interpolant B et le polynôme $A = (X-u_0)\cdots(X-u_{n-1})$. On exécute l'algorithme d'Euclide étendu sur A et B : on note R_j, U_j, V_j le reste et les cofacteurs obtenus à la j-ième itération. On choisit ensuite j minimal tel que $\deg(R_j) < k$. Si $\operatorname{pgcd}(A, V_j) = 1$, alors R_j/V_j est l'unique forme canonique de la fraction rationnelle cherchée. Sinon, le problème (7.4) n'a pas de solution.

Une application importante de l'interpolation rationnelle est la reconnaissance de récurrences linéaires d'ordre 1 à coefficients polynomiaux.

Corollaire 7.4 — Devinette de récurrences hypergéométriques. Soit (a_n) une suite d'éléments non nuls de $\mathbb{K}$ vérifiant la récurrence $p_1(n)a_{n+1} + p_0(n)a_n = 0$, dont les coefficients (inconnus) $p_0(X)$ et $p_1(X)$ sont des polynômes de $\mathbb{K}[X]$, premiers entre eux (récurrence hypergéométrique). Si l'on connaît une borne d sur le degré de p_0 et de p_1 telle que $p_0(j) \neq 0$ pour $0 \leq j \leq 2d+1$, alors on peut déterminer p_0 et p_1 par un calcul d'interpolation rationnelle, à partir des $2d+1$ premiers termes de la suite (a_n).

Algorithmes rapides

En combinant les résultats du Théorème 7.2 ci-dessus, du Théorème 6.16 en page 134, et du Chapitre 5 pour le calcul rapide des polynômes A et B définis en Section 7.1, on déduit le résultat de complexité suivant.

Théorème 7.5 Soient $A \in \mathbb{K}[X]$ de degré $n > 0$, $B \in \mathbb{K}[X]$ de degré $< n$ et $k \in \{1,2,\dots,n\}$. Soient $v_0,\dots,v_{n-1} \in \mathbb{K}$, et soient $u_0,\dots,u_{n-1}$ des points distincts de $\mathbb{K}$. Il est possible de calculer en $O(M(n)\log n)$ opérations dans $\mathbb{K}$:

1. une solution du problème reconstruction rationnelle (RR) ;
2. un approximant de Padé pour B de type $(k-1, n-k)$;
3. une fraction rationnelle $F = R/V \in \mathbb{K}(X)$ telle que $\deg(V) \leq n-k$, $\deg(R) < k$, et $F(u_i) = v_i$ pour $0 \leq i < n$.

Le cas entier

L'analogue pour les entiers de la reconstruction rationnelle est également calculable en complexité quasi-optimale. Si n et B sont deux entiers, $0 \leq B < n$, et étant donné $k \leq n$, il s'agit de calculer des entiers R et V tels que

$$\operatorname{pgcd}(n, V) = 1 \quad \text{et} \quad B = \frac{R}{V} \bmod n, \quad |R| < k, \quad 0 \leq V \leq \frac{n}{k}. \tag{7.5}$$

Théorème 7.6 À partir de k, n et B, on peut calculer R et V satisfaisant à (7.5) en $O(M_{\mathbb{Z}}(\log n)\log\log n)$ opérations binaires.

Le cas des récurrences

Une conséquence utile du Théorème 7.5 à la reconnaissance rapide des suites est contenue dans le corollaire suivant. Sa généralisation aux cas des suites polynomialement récurrentes arbitraires sera traitée en Section 7.2.

Corollaire 7.7 Soit (a_n) une suite d'éléments de $\mathbb{K}$ vérifiant
- soit une récurrence linéaire à coefficients constants d'ordre au plus N ;
- soit une récurrence linéaire d'ordre 1 à coefficients polynomiaux de degré au plus N.

À partir des premiers $2N+1$ termes de la suite, on peut retrouver les coefficients de la récurrence en $O(M(N)\log N)$ opérations dans $\mathbb{K}$.

7.2 Approximants de Padé–Hermite

Définition 7.2 Soit $\mathbf{F} = {}^t(f_1, \dots, f_n)$ un vecteur de $n \geq 1$ séries formelles de $\mathbb{K}[[X]]$, et soit $\mathbf{d} = (d_1, \dots, d_n) \in \mathbb{N}^n$. Un vecteur non nul $\mathbf{P} = (P_1, \dots, P_n)$ de polynômes de $\mathbb{K}[X]$ est appelé *approximant de Padé–Hermite de type* $\mathbf{d}$ *de* $\mathbf{F}$ si :
1. La valuation $\mathrm{val}(\mathbf{P}\cdot\mathbf{F})$ de la série $\mathbf{P}\cdot\mathbf{F} = \sum_{i=1}^{n} P_i f_i$ est au moins égale à $\sigma = \sum(d_i+1) - 1$;
2. $\deg(P_i) \leq d_i$ pour tout $1 \leq i \leq n$.

L'entier σ est alors appelé l'*ordre de l'approximant*.

Exemple 7.1 Dans la terminologie de la section précédente, si $R/V \in \mathbb{K}(X)$ est un approximant de Padé de type $(k, n-k)$ de $B \in \mathbb{K}[[X]]$, alors (R, V) est un approximant de Padé–Hermite pour $(-1, B)$, de type $(k, n-k)$.

Exercice 7.1 Soient A, B deux polynômes de $\mathbb{K}[X]$, avec $n = \deg(A) > \deg(B)$. Montrer que (R, V) est solution du problème de reconstruction rationnelle (RRS) défini en page 140 si et seulement s'il existe un polynôme $U \in \mathbb{K}[X]$ tel que (R, V, U) soit un approximant de Padé–Hermite de $(-1, B, A)$ de type $(k-1, n-k, n-1)$. ∎

Le premier résultat de cette section montre l'existence des approximants de Padé–Hermite et fournit également un premier algorithme pour leur calcul.

Théorème 7.8 Tout vecteur de séries formelles $\mathbf{F} = (f_1, \dots, f_n) \in \mathbb{K}[[X]]^n$ admet un approximant de Padé–Hermite de type $\mathbf{d} = (d_1, \dots, d_n) \in \mathbb{N}^n$ donné.

Démonstration. On procède par coefficients indéterminés : en écrivant $P_i = \sum_{j=0}^{d_i} p_{i,j} X^j$, on obtient un système linéaire homogène à $\sigma = \sum_i (d_i+1) - 1$ équations en les $\sigma + 1$

inconnues $p_{i,j}$. Puisqu'il a moins d'équations que d'inconnues, ce système admet forcément une solution non triviale. ■

L'algorithme qui découle de la preuve du Théorème 7.8 repose sur la recherche d'un élément non trivial dans le noyau d'une matrice à coefficients dans $\mathbb{K}$, de taille $\sigma \times (\sigma+1)$. En utilisant les résultats du Chapitre 8, on aboutit donc à un algorithme de complexité $O(\mathsf{MM}(\sigma)) = O(\sigma^\theta)$ pour $2 \le \theta \le 3$.

Algorithme de Derksen : idées et résultats préliminaires

Le but de la suite de ce chapitre est de présenter un algorithme plus efficace, dû à Derksen, de complexité seulement quadratique en σ. En anticipant un peu, cet algorithme revient à effectuer une sorte de pivot de Gauss sur une matrice à coefficients dans $\mathbb{K}[X]$, mais de taille bien plus petite que σ (seulement linéaire en $\max(\mathbf{d})$). Dans le cas particulier $n = 2$, l'algorithme de Derksen a essentiellement la même complexité que le calcul d'approximants de Padé via l'algorithme d'Euclide étendu vu en Section 7.1.

Pour simplifier la présentation, on se restreint dans la suite au cas où le type de l'approximant cherché est de la forme $\mathbf{d} = (d, \dots, d) \in \mathbb{N}^n$, pour un certain $d \in \mathbb{N}$. L'idée de l'algorithme de Derksen est de construire non pas un seul approximant de Padé–Hermite, mais toute une famille de tels approximants, et cela de manière incrémentale. Plus exactement, pour $s = 0, 1, \dots$, il construit une base d'une forme spéciale, appelée « base minimale », du $\mathbb{K}[X]$-module

$$V_s = \{\mathbf{P} \in \mathbb{K}[X]^n \mid \mathrm{val}(\mathbf{P} \cdot \mathbf{F}) \ge s\}.$$

Comme nous le verrons plus loin, une telle base de V_σ pour $\sigma = nd + n - 1$ contiendra alors nécessairement un approximant de Padé–Hermite de type $\mathbf{d} = (d, \dots, d)$ de $\mathbf{F}$.

Observons que, grâce aux inclusions $X^s\mathbb{K}[X]^n \subseteq V_s \subseteq \mathbb{K}[X]^n$, le module V_s est libre de rang n. Précisons ce que l'on entend par « base minimale ». Pour ce faire, introduisons d'abord une notion de *degré* et de *type* d'un vecteur de polynômes.

Définition 7.3 — Degré et type d'un vecteur de polynômes. Pour un vecteur de polynômes $\mathbf{P} = (P_1, \dots, P_n) \in \mathbb{K}[X]^n$, son *degré* $\deg(\mathbf{P})$ et son *type* $\mathrm{type}(\mathbf{P})$ sont donnés par :

$$\deg(\mathbf{P}) = \max\{\deg(P_1), \dots, \deg(P_n)\} \quad \text{et} \quad \mathrm{type}(\mathbf{P}) = \max\{i \mid \deg(\mathbf{P}) = \deg(P_i)\}.$$

Définition 7.4 — Base minimale. Soit $V \subseteq \mathbb{K}[X]^n$ un sous-module libre de rang n. Une suite $\mathbf{Q}_1, \dots, \mathbf{Q}_n$ est appelée « base minimale » de V si, pour tout i, le vecteur $\mathbf{Q}_i$ est non nul, de type i, et de degré minimal parmi les éléments de $V \setminus \{0\}$ de type i.

Le résultat suivant précise le lien entre une base minimale et l'approximation de Padé–Hermite.

Lemme 7.9 Soit $d \geq 1$. Supposons que $\mathbf{Q}_1, \ldots, \mathbf{Q}_n$ est une base minimale de V_{nd+n-1}. Soit ℓ tel que $\deg(\mathbf{Q}_\ell)$ est minimal. Alors $\mathbf{Q}_\ell$ est un approximant de Padé–Hermite de type $(d, \ldots, d)$ pour $\mathbf{F}$.

Démonstration. Le Théorème 7.8 montre l'existence d'un vecteur non nul $\mathbf{P}$ de V_{nd+n-1} tel que $\deg(\mathbf{P}) \leq d$. Soit i le type de $\mathbf{P}$. La suite d'inégalités

$$\deg(\mathbf{Q}_\ell) \leq \deg(\mathbf{Q}_i) \leq \deg(\mathbf{P}) \leq d,$$

prouve que $\mathbf{Q}_\ell$ est un approximant de Padé–Hermite de type $(d, \ldots, d)$ de $\mathbf{F}$. ■

Le résultat suivant montre qu'une base minimale est nécessairement une base du $\mathbb{K}[X]$-module V au sens usuel.

Théorème 7.10 Soit $\mathrm{V} \subseteq \mathbb{K}[X]^n$ un sous-module libre de rang n. Toute base minimale de V est une base du $\mathbb{K}[X]$-module V.

Démonstration. Montrons d'abord qu'il s'agit d'un système de générateurs. Soit $\mathrm{W} = \mathbb{K}[X]\mathbf{Q}_1 + \cdots + \mathbb{K}[X]\mathbf{Q}_n \subseteq \mathrm{V}$. On suppose par l'absurde que $\mathrm{V} \neq \mathrm{W}$. Soit $\mathbf{P} \in \mathrm{V} \setminus \mathrm{W}$ un élément minimal dans $\mathrm{V} \setminus \mathrm{W}$ pour l'ordre $\mathbf{P} < \mathbf{Q}$ défini par

$$\begin{aligned} &\deg(\mathbf{P}) < \deg(\mathbf{Q}), \quad \text{ou} \\ &\deg(\mathbf{P}) = \deg(\mathbf{Q}) \quad \text{et} \quad \mathrm{type}(\mathbf{P}) < \mathrm{type}(\mathbf{Q}). \end{aligned} \tag{7.6}$$

Autrement dit, $\mathbf{P}$ est de type minimal parmi les éléments de degré minimal de $\mathrm{V} \setminus \mathrm{W}$.

Soit i le type de $\mathbf{P}$. Les relations $\mathrm{type}(\mathbf{P}) = \mathrm{type}(\mathbf{Q}_i)$ et $\deg(\mathbf{P}) \geq \deg(\mathbf{Q}_i)$ entraînent l'existence d'un monôme $q \in \mathbb{K}[X]$ de degré $\deg(q) = \deg(\mathbf{P}) - \deg(\mathbf{Q}_i)$ tel que $\mathrm{type}(\mathbf{P} - q\mathbf{Q}_i) < \mathrm{type}(\mathbf{P})$. Puisque $\deg(\mathbf{P} - q\mathbf{Q}_i) \leq \deg(\mathbf{P})$, on obtient que

$$\mathbf{P} - q\mathbf{Q}_i < \mathbf{P}.$$

Par la minimalité de $\mathbf{P}$, il s'ensuit que $\mathbf{P} - q\mathbf{Q}_i$ appartient à W, donc $\mathbf{P} \in \mathrm{W}$, ce qui contredit le choix de $\mathbf{P}$.

Pour conclure la preuve, montrons que les $\mathbf{Q}_i$ forment une famille libre. Si $\sum_i a_i\mathbf{Q}_i = 0$ est une combinaison polynomiale nulle des $\mathbf{Q}_i$, on a que pour tout i, le vecteur $a_i\mathbf{Q}_i$ est de type i. L'assertion découle du lemme suivant. ■

Lemme 7.11 Si $\mathbf{P}$ et $\mathbf{Q}$ ont des types distincts, alors le type de $\mathbf{P} + \mathbf{Q}$ est soit le type de $\mathbf{P}$ soit celui de $\mathbf{Q}$.

Démonstration. Supposons $j = \mathrm{type}(\mathbf{P}) > i = \mathrm{type}(\mathbf{Q})$. Si $\deg(\mathbf{P}) \geq \deg(\mathbf{Q})$, alors on a $\mathrm{type}(\mathbf{P} + \mathbf{Q}) = j$ et si $\deg(\mathbf{P}) < \deg(\mathbf{Q})$, puis $\mathrm{type}(\mathbf{P} + \mathbf{Q}) = i$. ■

Algorithme de Derksen : fonctionnement

L'idée de l'algorithme est de construire de proche en proche une base minimale de V_s, partant de la base minimale des $\mathbf{Q}_k = (0,0,\dots,0,1,0,\dots,0)$ (avec 1 en position k) de V_0. D'après le Lemme 7.9, l'élément de degré minimal dans une base minimale de V_{nd+n-1} fournit un approximant de Padé–Hermite de type $(d,\dots,d)$ de $\mathbf{F}$.

Le résultat suivant montre comment construire une base minimale de V_{s+1} à partir d'une base minimale de V_s, et il constituera le cœur de l'itération.

Théorème 7.12 Soit $\mathbf{Q}_1,\dots,\mathbf{Q}_n$ une base minimale de

$$V_s = \{\mathbf{P} \in \mathbb{K}[X]^n \mid \mathrm{val}(\mathbf{P}\cdot\mathbf{F}) \geq s\}.$$

1. Si $\mathrm{val}(\mathbf{Q}_i\cdot\mathbf{F}) \geq s+1$ quel que soit i, alors V_{s+1} et V_s coïncident, et $\{\mathbf{Q}_1,\dots,\mathbf{Q}_n\}$ est une base minimale de V_{s+1}.
2. Supposons que $1 \leq i \leq n$ est tel que les deux conditions suivantes soient réunies :
   — $\mathrm{val}(\mathbf{Q}_i\cdot\mathbf{F}) = s$;
   — si $\mathrm{val}(\mathbf{Q}_\ell\cdot\mathbf{F}) = s$ pour un $\ell \neq i$, alors $\mathbf{Q}_i < \mathbf{Q}_\ell$, où $<$ est l'ordre (7.6).

   Alors :
   a. pour $\ell \neq i$, il existe un scalaire $\lambda_\ell \in \mathbb{K}$ tel que $\tilde{\mathbf{Q}}_\ell = \mathbf{Q}_\ell - \lambda_\ell\mathbf{Q}_i$ vérifie $\mathrm{val}(\tilde{\mathbf{Q}}_\ell\cdot\mathbf{F}) > s$;
   b. en posant $\tilde{\mathbf{Q}}_i = X\mathbf{Q}_i$, la suite $\tilde{\mathbf{Q}}_1,\dots,\tilde{\mathbf{Q}}_n$ forme une base minimale de V_{s+1}.

Démonstration. L'inclusion $V_{s+1} \subseteq V_s$ est évidente, et inversement, le Théorème 7.10 montre que tout $\mathbf{P} \in V_s$ s'écrit comme combinaison linéaire $\sum_i a_i\mathbf{Q}_i$. Ainsi $\mathbf{P}\cdot\mathbf{F} = \sum_i a_i(\mathbf{Q}_i\cdot\mathbf{F})$ est de valuation au moins $s+1$, donc $\mathbf{P} \in V_{s+1}$. Ceci prouve le premier point.

Prouvons maintenant l'assertion (2.a). Si $\mathrm{val}(\mathbf{Q}_\ell\cdot\mathbf{F}) > s$, on pose $\lambda_\ell = 0$; si $\mathrm{val}(\mathbf{Q}_\ell\cdot\mathbf{F}) = s$, alors $\mathbf{Q}_\ell\cdot\mathbf{F} = c_\ell X^s + \cdots$ et $\mathbf{Q}_i\cdot\mathbf{F} = c_i X^s + \cdots$, avec $c_i \neq 0$, et alors $\lambda_\ell = c_\ell/c_i$ convient.

Pour l'assertion (2.b), commençons par montrer que

$$\tilde{\mathbf{Q}}_1,\dots,\tilde{\mathbf{Q}}_{i-1},\mathbf{Q}_i,\tilde{\mathbf{Q}}_{i+1},\dots,\tilde{\mathbf{Q}}_n$$

reste une base minimale de V_s. Il suffit pour cela de montrer que pour $\ell \neq i$, le vecteur $\tilde{\mathbf{Q}}_\ell$ a même type et même degré que $\mathbf{Q}_\ell$. Si $\mathrm{val}(\mathbf{Q}_\ell\cdot\mathbf{F}) > s$, cela est évident, car $\lambda_\ell = 0$ et donc $\tilde{\mathbf{Q}}_\ell = \mathbf{Q}_\ell$. Sinon, le choix de i assure que $\mathbf{Q}_\ell > \mathbf{Q}_i$, et donc $\mathbf{Q}_\ell$ et $\mathbf{Q}_\ell - \lambda_\ell\mathbf{Q}_i$ ont même degré et même type.

Montrons maintenant que $(\tilde{\mathbf{Q}}_1,\dots,\tilde{\mathbf{Q}}_n)$ est une base minimale de V_{s+1}. Comme la multiplication par un polynôme ne change pas le type, celui de $\tilde{\mathbf{Q}}_i = X\mathbf{Q}_i$ est bien i. Il suffit donc de montrer que si $\mathbf{P} \in V_{s+1}$ est de type $\ell \in \{1,2,\dots,n\}$, alors $\deg(\mathbf{P}) \geq \deg(\mathbf{Q}_\ell)$. Si $\ell \neq i$, ceci est une évidence : comme $\mathbf{P}$ appartient à $V_{s+1} \subseteq V_s$, et comme la suite $\tilde{\mathbf{Q}}_1,\dots,\tilde{\mathbf{Q}}_{i-1},\mathbf{Q}_i,\tilde{\mathbf{Q}}_{i+1},\dots,\tilde{\mathbf{Q}}_n$ forme une base minimale de V_s, le degré de $\mathbf{P}$ est nécessairement au moins égal à $\deg(\tilde{\mathbf{Q}}_\ell) = \deg(\mathbf{Q}_\ell)$.

Dans la suite de la preuve, on peut donc supposer que $\mathbf{P} \in V_{s+1}$ est de type i, le but étant de montrer que $\deg(\mathbf{P}) \geq \deg(X\mathbf{Q}_i)$. Le Théorème 7.10 montre que $\mathbf{P}$ s'écrit

Entrée $\mathbf{F} = (f_1, \ldots, f_n) \in \mathbb{K}[[X]]^n$ et $d \geq 1$.
Sortie Un approximant de Padé–Hermite de $\mathbf{F}$, de type $(d, \ldots, d)$.

1. Pour k de 1 à n définir
   $\mathbf{Q}_k = (0, 0, \ldots, 0, 1, 0, \ldots, 0)$, avec 1 en position k.
2. Pour j de 0 à $nd + n - 2$ faire :
   a. $i = 0$
   b. Pour k de 1 à n faire :
      i. $c_k = \text{coeff}(\mathbf{Q}_k \cdot \mathbf{F}, j)$
      ii. Si $c_k \neq 0$ et ($\mathbf{Q}_k < \mathbf{Q}_i$ ou $i = 0$), alors $i = k$.
   c. Si $i \neq 0$ alors faire
      i. $\mathbf{Q}_i = c_i^{-1}\mathbf{Q}_i$
      ii. Pour k de 1 à n faire
         si $k \neq i$, alors $\mathbf{Q}_k = \mathbf{Q}_k - c_k\mathbf{Q}_i$.
      iii. $\mathbf{Q}_i = X\mathbf{Q}_i$
3. $p = 1$
4. Pour k de 2 à n faire :
   si $\deg(\mathbf{Q}_k) < \deg(\mathbf{Q}_p)$, alors $p = k$.
5. Renvoyer $\mathbf{Q}_p$.

Algorithme 7.3 – Algorithme de Derksen pour les approximants de Padé–Hermite.

$\mathbf{P} = \sum_{j \neq i} a_j \tilde{\mathbf{Q}}_j + a_i \mathbf{Q}_i$. Puisque $\text{type}(\mathbf{P}) = i$, le Lemme 7.11 entraîne $a_i \neq 0$. De plus, le degré de $\mathbf{P}$ est égal à celui de $a_i\mathbf{Q}_i$, d'après le Lemme 7.13 ci-dessous.

Comme $\text{val}(\mathbf{P} \cdot \mathbf{F}) > s$ et comme pour $k \neq i$, $\text{val}(\mathbf{Q}_k \cdot \mathbf{F}) > s$, on a nécessairement que $\text{val}(a_i) > 0$. En particulier $\deg(a_i) > 0$ et donc $\deg(\mathbf{P}) \geq 1 + \deg(\mathbf{Q}_i)$. ■

Lemme 7.13 Si $\text{type}(\mathbf{P}) = \text{type}(\mathbf{Q}) = i$ et $\text{type}(\mathbf{P} + \mathbf{Q}) < i$, alors $\deg(\mathbf{P}) = \deg(\mathbf{Q})$.

Démonstration. Soient $\mathbf{P} = (P_1, \ldots, P_n)$ et $\mathbf{Q} = (Q_1, \ldots, Q_n)$. L'hypothèse entraîne les égalités $\deg(P_i) = \deg(\mathbf{P})$ et $\deg(Q_i) = \deg(\mathbf{Q})$. Cela implique que P_i et Q_i ont le même degré ; sinon, le type de $\mathbf{P} + \mathbf{Q}$ serait égal à i. ■

L'Algorithme 7.3 découle de la conjonction du Lemme 7.9 et du Théorème 7.12. Il est de complexité seulement quadratique en σ. En résumé, nous avons le résultat suivant.

Théorème 7.14 Soit $\mathbf{F} = (f_1, \ldots, f_n) \in \mathbb{K}[[X]]^n$. Il est possible de calculer un approximant de Padé–Hermite de type $(d, \ldots, d)$ de $\mathbf{F}$ en $O(n\sigma^2) = O(n^3d^2)$ opérations dans $\mathbb{K}$.

Exercice 7.2 Écrire les détails de la preuve du Théorème 7.14. ■

R Dans le cas « générique » (dit *normal*), la sortie $\mathbf{Q} = \mathbf{Q}_1, \ldots, \mathbf{Q}_n$ de l'algorithme de Derksen est de degré

$$\begin{pmatrix} d+1 & d & \cdots & d & d \\ d+1 & d+1 & \cdots & d & d \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ d+1 & d+1 & \cdots & d+1 & d \\ d & d & \cdots & d & d \end{pmatrix}.$$

C'est la dernière ligne $\mathbf{Q}_n$ qui fournit l'approximant désiré.

Approximants de Padé–Hermite de type arbitraire

Pour calculer un approximant de Padé–Hermite de type $\mathbf{d} = (d_1, \ldots, d_n)$, il suffit de remplacer, dans l'Algorithme 7.3, deg par $\deg_{\mathbf{d}}$ et type par $\mathrm{type}_{\mathbf{d}}$, où :

$$\deg_{\mathbf{d}}(\mathbf{P}) = \max_{1 \le i \le n} \{\deg(\mathrm{P}_i) - d_i\}, \quad \mathrm{type}_{\mathbf{d}}(\mathbf{P}) = \max\{i \mid \deg(\mathrm{P}_i) - d_i = \deg_{\mathbf{d}}(\mathbf{P})\}.$$

Applications

Si l'on sait qu'une série est rationnelle de degré au plus d, l'approximation de Padé de type (d, d) suffit pour reconstruire la fraction rationnelle. Une question naturelle est de généraliser cette observation dans le cadre de l'approximation de Padé–Hermite. Les réponses sont partielles, mais entièrement satisfaisantes dans la pratique.

Recherche de relations à coefficients polynomiaux entre séries formelles

Supposons qu'il existe une combinaison linéaire $\sum_{i=1}^{n} \mathrm{P}_i f_i = 0$, à coefficients des polynômes $\mathrm{P}_i(\mathrm{X}) \in \mathbb{K}[\mathrm{X}]$ de degrés au plus d. Par ailleurs, supposons calculé un approximant de Padé–Hermite $\mathbf{Q} = (\mathrm{Q}_1, \ldots, \mathrm{Q}_n)$ de $\mathbf{F} = (f_1, \ldots, f_n)$ de type $\mathbf{d} = (d, \ldots, d)$, via l'algorithme de Derksen. La question est : quel lien y a-t-il entre $\mathbf{P} = (\mathrm{P}_1, \ldots, \mathrm{P}_n)$ et $\mathbf{Q}$?

D'abord, dans le cas *générique*, la réponse est très simple : $\mathbf{P}$ et $\mathbf{Q}$ sont identiques, à un coefficient scalaire près. En effet, $\mathbf{Q} = \mathbf{Q}_n$ et $\mathbf{Q}_1, \ldots, \mathbf{Q}_n$ est une base de V_{nd+n-1} dont les degrés sont décrits page 150. En particulier, $\mathbf{P}$ doit être une combinaison linéaire des $\mathbf{Q}_i$. Des considérations sur les degrés, utilisant la forme bien particulière des degrés des $\mathbf{Q}_i$, mènent à la conclusion désirée.

En effet, si $(c_1(\mathrm{X}), \ldots, c_n(\mathrm{X})) \cdot {}^t(\mathbf{Q}_1, \ldots, \mathbf{Q}_n) = \mathbf{P}$, alors $c_1 \mathrm{Q}_{11} + \cdots + c_n \mathrm{Q}_{n1} = \mathrm{P}_1$, et comme $\deg(\mathrm{Q}_{11}) = d + 1$ et $\deg(\mathrm{Q}_{j1}) = d$ pour $j > 1$ et $\deg(\mathrm{P}_1) \le d$, on obtient que $c_1 = 0$. De même, $c_2 = \cdots = c_{n-1} = 0$ et c_n doit être une constante $c \in \mathbb{K}$ telle que $\mathbf{P} = c\mathbf{Q}$.

Dans le cas général, un argument de noethérianité permet de prouver que pour $\mathrm{D} \gg 0$, $\mathrm{V}_{n\mathrm{D}+n-1}$ contient la relation $\mathbf{P}$, qui sera trouvée par l'algorithme de Derksen. Seulement, on ne dispose pas de borne *a priori* en fonction de d, sur le D minimal avec cette propriété. En effet, si on note

$$\mathrm{W}_j = \{\mathbf{Q} \in \mathbb{K}[\mathrm{X}]^n \mid \mathrm{val}(\mathbf{Q} \cdot \mathbf{F}) \ge j \quad \text{et} \quad \deg(\mathbf{Q}) \le \deg(\mathbf{P})\},$$

et $\mathrm{W}_\infty = \cap_{j \ge 0} \mathrm{W}_j$, alors W_∞ contient toutes les relations de F en degré d, et en particulier $\mathbf{P}$. Puisque $\mathrm{W}_0 \supseteq \mathrm{W}_1 \supseteq \mathrm{W}_2 \supseteq \cdots$ est une suite décroissante d'espaces vectoriels de dimension finie, elle est stationnaire, donc il existe un N tel que $\mathrm{W}_\infty = \mathrm{W}_\mathrm{N} = \mathrm{W}_{\mathrm{N}+1} = \cdots$.

Le cas *normal* correspond à la situation où $\dim(W_{k+1}) = \dim(W_k) - 1$ pour chaque k (noter la ressemblance avec la normalité de la suite des restes dans l'algorithme d'Euclide).

Reconstruction d'équations algébriques et différentielles

Deux cas particuliers importants, pour lesquels on peut être encore plus précis, sont les approximants algébriques et différentiels.

Soit $A \in \mathbb{K}[[X]]$ une série formelle algébrique. Le problème d'approximation algébrique consiste à retrouver un polynôme $P(X,Y) \in \mathbb{K}[X,Y]$ tel que $P(X,A(X)) = 0$ à partir des premiers termes de A. Si un tel P, de degré d en X et n en Y, existe, alors les coefficients des puissances de Y formeront un approximant de Padé–Hermite de type $(d,\dots,d)$ du vecteur de séries $(1,A,\dots,A^n)$. La difficulté vient de ce que un calcul d'approximation de Padé–Hermite ne trouve, *a priori*, qu'un polynôme $Q \in \mathbb{K}[X,Y]$ tel que $Q(X,A(X)) = 0 \bmod X^\sigma$, où $\sigma = (n+1)d - 1$. On dit alors qu'on a *deviné* un *polynôme annulateur* Q de A. Pour les approximants différentiels, la problématique est la même, la seule différence étant qu'on calcule un approximant de Padé–Hermite du vecteur des dérivées successives $(A, A', \dots, A^{(n)})$.

Certification d'identités algébriques

Pour un approximant algébrique, il se pose la question de la *certification a posteriori*. Pour un polynôme annulateur deviné Q, cela veut dire qu'on souhaiterait déduire que non seulement $Q(X,A(X))$ est nul modulo X^σ, mais aussi que $Q(X,A(X)) = 0$.

Le résultat suivant apporte une réponse partielle à la question de la certification. Son avantage est qu'il ne dépend pas de l'algorithme utilisé pour produire l'approximant de Padé–Hermite.

Théorème 7.15 Supposons que $A \in \mathbb{K}[[X]]$ est racine d'un polynôme irréductible de $\mathbb{K}[X,Y]$ de degré au plus d en X et au plus n en Y. Soit

$$\mathbf{Q} = (Q_0, Q_1, \dots, Q_n)$$

un approximant de Padé–Hermite de type $(d,\dots,d)$ de

$$\mathbf{F} = (1, A, \dots, A^n).$$

Si $\mathrm{val}(\mathbf{Q}\cdot\mathbf{F}) \geq 2dn$, alors $\mathbf{Q}\cdot\mathbf{F} = 0$, c'est-à-dire que A est racine du polynôme $Q = \sum_{i=0}^{n} Q_i Y^i$.

Démonstration. Soit $P \in \mathbb{K}[X,Y]$ un polynôme irréductible de degré au plus d en X et au plus n en Y tel que $P(X,A) = 0$. Le polynôme $\mathrm{Res}_Y(P,Q) \in \mathbb{K}[X]$ est de degré borné par $\deg_X(P)\deg_Y(Q) + \deg_Y(P)\deg_X(Q) \leq 2dn$. Comme il est de la forme $uP + vQ$ pour u et v deux polynômes, avec $\deg_Y(v) < \deg_Y(P)$, et qu'il ne change pas lorsqu'on remplace la variable Y par la série A, c'est un $O(X^{2dn})$. Par conséquent $uP + vQ = 0$, donc P divise vQ et, étant irréductible, il divise v ou Q. Or $\deg_Y(v) < \deg_Y(P)$, donc finalement $Q \mid P$ et $Q(X,A)$ est la série nulle. ■

Reconstruction de récurrences

Soit $(a_n)_n \in \mathbb{K}^{\mathbb{N}}$ une suite vérifiant une récurrence linéaire à coefficients polynomiaux. Comment, à partir des premiers termes de la suite, retrouver les coefficients de cette récurrence ?

Une possibilité consiste à utiliser la correspondance montrée au Théorème 14.1 en page 253, et à chercher une équation différentielle, à coefficients polynomiaux, portant sur la série génératrice de la suite. Cette équation peut être devinée comme expliqué précédemment, grâce à une approximation de Padé–Hermite. Il suffit ensuite de passer de l'équation différentielle à l'expression de la récurrence, ce qui n'est pas trivial mais purement formel et a été décrit au Chapitre 14.

Calcul quasi-optimal

Pour le calcul d'approximants de Padé–Hermite, on dispose d'un algorithme quasi-optimal, dont l'efficacité est utilisée en calcul formel comme la base de nombreux autres algorithmes. Cet algorithme peut être vu comme une alternative à l'approche par *matrices structurées*, de même complexité, décrite au Chapitre 10.

Théorème 7.16 Soient $\mathbf{F} = (f_1, \dots, f_n) \in \mathbb{K}[[X]]^n$, $\mathbf{d} = (d_1, \dots, d_n) \in \mathbb{N}^n$ et $\sigma = d_1 + \cdots + d_n + n - 1$. Il est possible de calculer un approximant de Padé–Hermite de type $\mathbf{d}$ de $\mathbf{F}$ en $O(\mathsf{MM}(n,\sigma)\log\sigma) = O(n^\theta \mathsf{M}(\sigma)\log\sigma)$ opérations dans $\mathbb{K}$.

Esquisse de démonstration. Comme pour l'algorithme de complexité quadratique de Derksen, une première idée est de calculer non pas un vecteur, mais toute une matrice d'approximants. Une seconde idée est d'utiliser un « diviser pour régner » tout en exploitant la multiplication rapide de matrices polynomiales. Voici l'esquisse de l'algorithme dans le cas où $\mathbf{d} = (d, \dots, d)$:

1. Si $N = 1$, renvoyer une base de l'orthogonal à $\mathbf{F}(0)$.
2. Sinon :
   (a) calculer récursivement une matrice polynomiale d'approximants $\mathbf{P}_1$ telle que $\mathbf{P}_1 \cdot \mathbf{F} = O(X^{\sigma/2})$ et $\deg(\mathbf{P}_1)$ vaille environ $d/2$;
   (b) calculer le reste $\mathbf{R}$ tel que $\mathbf{P}_1 \cdot \mathbf{F} = X^{\sigma/2} \cdot \left(\mathbf{R} + O(X^{\sigma/2})\right)$;
   (c) calculer récursivement une matrice polynomiale $\mathbf{P}_2$ d'approximants du reste : $\mathbf{P}_2 \cdot \mathbf{R} = O(X^{\sigma/2})$, avec $\deg(\mathbf{P}_2)$ environ égal à $d/2$;
   (d) renvoyer $\mathbf{P} = \mathbf{P}_2 \cdot \mathbf{P}_1$.

Le choix précis des degrés des approximants $\mathbf{P}_1$ et $\mathbf{P}_2$ est une question délicate et ne sera pas détaillé. En admettant ces choix, la correction de l'algorithme et son analyse de complexité se déduisent aisément. ■

Exercices

Exercice 7.3 — Interpolation creuse. Soit $\mathbb{K}$ un corps effectif de caractéristique zéro. Soit F une fonction d'arité n sur $\mathbb{K}$ qui calcule les valeurs d'un polynôme P de $\mathbb{K}[X_1, \dots, X_n]$ composé d'au plus t termes non nuls. Le calcul de tous les termes non nuls de P en évaluant seulement F sur plusieurs points bien choisis est appelé

interpolation creuse.

Soient $p_1, \ldots, p_n$ des nombres premiers distincts (en pratique on choisit les plus petits).

1. Soient $X_1^{e_1} \cdots X_n^{e_n}$ et $X_1^{f_1} \cdots X_n^{f_n}$ deux monômes distincts. Expliquer brièvement pourquoi leurs valeurs en $(p_1, \ldots, p_n)$ sont distinctes.
2. Supposons que P contienne exactement $\tau \leq t$ termes non nuls, et qu'il s'écrive sous la forme $\sum_{i=1}^{\tau} P_i X_1^{e_{i,1}} \cdots X_n^{e_{i,n}}$. Montrer l'identité suivante :

$$\sum_{i \geq 0} F(p_1^i, \ldots, p_n^i) t^i = \sum_{i=1}^{\tau} \frac{P_i}{1 - p_1^{e_{i,1}} \cdots p_n^{e_{i,n}} t}.$$

3. Déduire de la formule précédente un algorithme pour obtenir les coefficients du polynôme $D(t) = \prod_{i=1}^{\tau} (1 - p_1^{e_{i,1}} \cdots p_n^{e_{i,n}} t)$ de $\mathbb{Q}[t]$. Estimer le coût de l'algorithme en fonction du nombre d'appels à la fonction F et du nombre d'opérations arithmétiques dans $\mathbb{K}$.
4. En admettant que l'on dispose d'un algorithme pour calculer toutes les racines de D dans $\mathbb{Q}$, expliquer comment obtenir toutes les valeurs $e_{i,j}$, $1 \leq i \leq \tau$, $1 \leq j \leq n$.
5. En utilisant les algorithmes rapides du Chapitre 5, et les racines de D, proposer une façon efficace de calculer tous les coefficients P_i de P.

■

Notes

La présentation de la Section 7.1 est fortement inspirée de l'article de McEliece et Shearer [MS78], celle de la Section 7.2 par l'article de Derksen [Der94]. Les approximants de Padé sont des objets très classiques en analyse numérique ; on pourra par exemple consulter le livre de Baker et Graves–Morris [BG96] qui leur est entièrement dédié.

L'algorithme de Berlekamp–Massey a été initialement conçu pour décoder des codes BCH, puis étendu par Zierler [Zie68] et Massey [Mas69], qui ont mis en évidence le lien avec les suites récurrentes linéaires à coefficients constants. Les liens forts entre l'algorithme d'Euclide, l'algorithme de Berlekamp–Massey, l'approximation de Padé et le calcul de fractions continues ont été étudiés par Mills [Mil75], Welch et Scholtz [WS79], Cheng [Che84], et Dornstetter [Dor87]. L'utilisation de ces méthodes pour l'interpolation creuse est à la base de l'algorithme dû à Ben-Or et Tiwari [BT88] adapté dans l'Exercice 7.3.

Les premiers algorithmes rapides pour la reconstruction rationnelle sont dus à Gustavson et Yun [GY79]. Ces algorithmes ont été améliorés par ces deux auteurs en collaboration avec Brent [BGY80] et Gragg–Warner [Gra+82], puis par Cabay et Choi [CC86]. La problématique de la reconstruction rationnelle pour les entiers est classique en théorie des nombres. En calcul formel, elle a été très étudiée en lien avec les algorithmes modulaires, par exemple pour la résolution de systèmes linéaires par remontées p-adiques par Dixon [Dix82]. Des algorithmes rapides ont été récemment conçus par Wang et Pan [PW04 ; WP03].

Les approximants de Padé et de Padé–Hermite ont été introduits et étudiés par

Hermite et Padé [Her73a ; Her93 ; Pad92 ; Pad94]. Une exposition plus moderne se trouve dans un article de Mahler [Mah68]. Le problème d'approximation de Padé–Hermite a été utilisé par Hermite en 1873 dans sa preuve [Her73b] de la transcendance de e, qui utilise le choix très particulier $\mathbf{F} = (1, e^X, e^{2X}, \ldots)$. Deux autres cas particuliers importants sont : les « approximants algébriques » [Sha74 ; TD00] avec $\mathbf{F} = (1, f, f^2, \ldots)$ et les « approximants différentiels » [Kha02] avec $\mathbf{F} = (f, f', f'', \ldots)$, où f est une série de $\mathbb{K}[[X]]$ donnée.

En calcul formel, les approximants de Padé–Hermite ont été introduits par Della Dora et Dicrescenzo [DD84a ; DD84b]. Un premier algorithme de complexité quadratique a été donné par Sergeyev [Ser86]. Cet article est resté méconnu, et d'autres algorithmes quadratiques ont été proposés par Paszkowski [Pas87], Cabay et Labahn [CL89], Van Barel et Bultheel [VB91b], et Cabay, Labahn, Beckermann [CLB92]. L'algorithme de complexité quasi-optimale esquissé en Section 7.2 a été conçu par Beckermann et Labahn [BL94].

Il existe diverses généralisations utiles du problème d'approximation de Padé–Hermite, par exemple les « approximants de Padé–Hermite simultanés et matriciels », ou encore des approximants modulo un polynôme arbitraire de degré $\sigma = \sum_i (d_i + 1) - 1$ au lieu de X^σ. Des algorithmes rapides existent pour toutes ces généralisations [Bec92 ; BL00 ; BL94 ; GJV03 ; VB92].

Bibliographie

Bec92 Beckermann, Bernhard (1992). « A reliable method for computing M-Padé approximants on arbitrary staircases ». In : *Journal of Computational and Applied Mathematics*, vol. 40, n°1, p. 19–42.

BG96 Baker Jr., George A. et Peter Graves-Morris (1996). *Padé approximants*. 2e éd. Vol. 59. Encyclopedia of Mathematics and its Applications. Cambridge University Press.

BGY80 Brent, Richard P., Fred G. Gustavson et David Y. Y. Yun (1980). « Fast solution of Toeplitz systems of equations and computation of Padé approximants ». In : *Journal of Algorithms*, vol. 1, n°3, p. 259–295.

BL00 Beckermann, Bernhard et George Labahn (2000). « Fraction-free computation of matrix rational interpolants and matrix GCDs ». In : *SIAM Journal on Matrix Analysis and Applications*, vol. 22, n°1, p. 114–144.

BL94 Beckermann, B. et G. Labahn (1994). « A uniform approach for the fast computation of matrix-type Padé approximants ». In : *SIAM Journal on Matrix Analysis and Applications*, vol. 15, n°3, p. 804–823.

BT88 Ben-Or, M. et P. Tiwari (1988). « A deterministic algorithm for sparse multivariate polynomial interpolation ». In : *STOC'88 : ACM Symposium on Theory of Computing*. ACM Press, p. 301–309.

CC86 Cabay, Stanley et Dong Koo Choi (1986). « Algebraic computations of scaled Padé fractions ». In : *SIAM Journal on Computing*, vol. 15, n°1, p. 243–270.

Che84 Cheng, Unjeng (1984). « On the continued fraction and Berlekamp's algorithm ». In : *IEEE Transactions on Information Theory*, vol. 30, n°3, p. 541–544.

CL89 Cabay, S. et G. Labahn (1989). « A fast, reliable algorithm for calculating Padé–Hermite forms ». In : *ISSAC'89 : International Symposium on Symbolic and Algebraic Computation*. Portland, Oregon, United States : ACM Press, p. 95–100.

CLB92 Cabay, S., G. Labahn et B. Beckermann (1992). « On the theory and computation of nonperfect Padé–Hermite approximants ». In : *Journal of Computational and Applied Mathematics*, vol. 39, n°3, p. 295–313.

DD84a Della Dora, J. et C. Dicrescenzo (1984). « Approximants de Padé–Hermite. I. Théorie ». In : *Numerische Mathematik*, vol. 43, n°1, p. 23–39.

DD84b — (1984). « Approximants de Padé–Hermite. II. Programmation ». In : *Numerische Mathematik*, vol. 43, n°1, p. 41–57.

Der94 Derksen, Harm (1994). *An algorithm to compute generalized Padé–Hermite forms*. Report n°9403. Dept. of Math., Catholic University Nijmegen.

Dix82 Dixon, John D. (1982). « Exact solution of linear equations using p-adic expansions ». In : *Numerische Mathematik*, vol. 40, n°1, p. 137–141.

Dor87 Dornstetter, Jean-Louis (1987). « On the equivalence between Berlekamp's and Euclid's algorithms ». In : *IEEE Transactions on Information Theory*, vol. 33, n°3, p. 428–431.

GJV03 Giorgi, Pascal, Claude-Pierre Jeannerod et Gilles Villard (2003). « On the complexity of polynomial matrix computations ». In : *ISSAC'03 : International Symposium on Symbolic and Algebraic Computation*. Éd. par J. R. Sendra. New York : ACM Press, p. 135–142.

Gra+82 Gragg, William B., Fred G. Gustavson, Daniel D. Warner et David Y. Y. Yun (1982). « On fast computation of superdiagonal Padé fractions ». In : *Mathematical Programming Study*, n°18. Algorithms and theory in filtering and control (Lexington, Ky., 1980), p. 39–42.

GY79 Gustavson, Fred G. et David Y. Y. Yun (1979). « Fast algorithms for rational Hermite approximation and solution of Toeplitz systems ». In : *IEEE Transactions on Circuits and Systems*, vol. 26, n°9, p. 750–755.

Her73a Hermite, C. (1873). « Extrait d'une lettre de Monsieur Ch. Hermite à Monsieur Paul Gordan ». In : *Journal für die Reine und Angewandte Mathematik*, vol. 78, p. 303–311.

Her73b — (1873). « Sur la fonction exponentielle ». In : *Comptes-rendus de l'Académie des sciences de Paris*, vol. 77, p. 18–24.

Her93 — (1893). « Sur la généralisation des fractions continues algébriques ». In : *Annali di Matematica pura ed applicata*, vol. 21, n°2, p. 289–308.

Kha02 Khan, M. A. H. (2002). « High-order differential approximants ». In : *Journal of Computational and Applied Mathematics*, vol. 149, n°2, p. 457–468.

Mah68 Mahler, K. (1968). « Perfect systems ». In : *Compositio Mathematica*, vol. 19, 95–166 (1968).

Mas69 Massey, James L. (1969). « Shift-register synthesis and BCH decoding ». In : *IEEE Transactions on Information Theory*, vol. IT-15, p. 122–127.

Mil75 Mills, W. H. (1975). « Continued fractions and linear recurrences ». In : *Mathematics of Computation*, vol. 29, p. 173–180.

MS78 McEliece, Robert J. et James B. Shearer (1978). « A property of Euclid's algorithm and an application to Padé approximation ». In : *SIAM Journal on Applied Mathematics*, vol. 34, n°4, p. 611–615.

Pad92 Padé, H. (1892). « Sur la représentation approchée d'une fonction par des fractions rationelles ». In : *Annales scientifiques de l'É.N.S.* vol. 9, p. 3–93.

Pad94 — (1894). « Sur la généralisation des fractions continues algébriques ». In : *Journal de mathématiques pures et appliquées*, vol. 10, n°4, p. 291–330.

Pas87 Paszkowski, Stefan (1987). « Recurrence relations in Padé–Hermite approximation ». In : *Journal of Computational and Applied Mathematics*, vol. 19, n°1, p. 99–107.

PW04 Pan, Victor Y. et Xinmao Wang (2004). « On rational number reconstruction and approximation ». In : *SIAM Journal on Computing*, vol. 33, n°2, p. 502–503.

Ser86 Sergeyev, A. V. (1986). « A recursive algorithm for Padé–Hermite approximations ». In : *USSR Comput. Maths Math. Phys*, vol. 26, n°2, p. 17–22.

Sha74 Shafer, R. E. (1974). « On quadratic approximation ». In : *SIAM Journal on Numerical Analysis*, vol. 11, p. 447–460.

TD00 Tourigny, Y. et Ph. G. Drazin (2000). « The asymptotic behaviour of algebraic approximants ». In : *The Royal Society of London. Proceedings. Series A. Mathematical, Physical and Engineering Sciences*, vol. 456, n°1997, p. 1117–1137.

VB91b Van Barel, Marc et Adhemar Bultheel (1991). « The computation of nonperfect Padé–Hermite approximants ». In : *Numerical Algorithms*, vol. 1, n°3, p. 285–304.

VB92 Van Barel, M. et A. Bultheel (1992). « A general module-theoretic framework for vector M-Padé and matrix rational interpolation ». In : *Numerical Algorithms*, vol. 3, n°1-4, p. 451–461.

WP03 Wang, Xinmao et Victor Y. Pan (2003). « Acceleration of Euclidean algorithm and rational number reconstruction ». In : *SIAM Journal on Computing*, vol. 32, n°2, p. 548–556.

WS79 Welch, L. R. et R. A. Scholtz (1979). « Continued fractions and Berlekamp's algorithm ». In : *IEEE Transactions on Information Theory*, vol. 25, n°1, p. 19–27.

Zie68 Zierler, N. (1968). « Linear recurring sequences and error-correcting codes ». In : *Error Correcting Codes (Proc. Sympos. Math. Res. Center, Madison, Wis., 1968)*. Wiley, p. 47–59.

II Matrices

8. Algèbre linéaire dense : de Gauss à Strassen

Résumé

L'algèbre linéaire est au cœur de nombreux problèmes algorithmiques. La multiplication matricielle usuelle et la méthode du pivot de Gauss ont une complexité arithmétique cubique en la taille de l'entrée. Dans ce chapitre, nous montrons que pour la plupart des questions d'algèbre linéaire dense — multiplication, inversion, déterminant, résolution de système — il existe des algorithmes plus efficaces, de complexité strictement sous-cubique.

8.1 Introduction

En mathématiques, il est de coutume de considérer qu'un problème est rendu trivial lorsqu'il est ramené à une question d'algèbre linéaire. Du point de vue calculatoire se pose cependant la question de l'efficacité des opérations matricielles.

Classification

Les problèmes algorithmiques en algèbre linéaire cachent des difficultés très subtiles. En général, les premières questions que l'on est amené à se poser en rapport avec la manipulation des matrices concernent le calcul efficace du produit matrice-matrice, du produit matrice-vecteur, de l'inverse, ainsi que la résolution de systèmes.

Les réponses à ces questions varient fortement selon le type de matrices considérées. Une classification possible est la suivante.

Les matrices denses

Ce sont les matrices quelconques, sans aucune structure particulière. Nous verrons que l'algorithmique des matrices denses peut se ramener essentiellement au produit matriciel, dont la complexité est une question extrêmement délicate.

Les matrices creuses

Beaucoup de problèmes linéaires se formulent en termes de matrices possédant un nombre important d'éléments nuls, dites « creuses ». Dans ce cas, l'algorithmique dense est inappropriée ; il est possible de tirer parti de la forme creuse en utilisant des outils mieux adaptés, à base de récurrences linéaires.

Les matrices structurées

Enfin, il existe des familles de matrices particulières, souvent associées à des applications linéaires entre espaces de polynômes, telles que les matrices de type Sylvester pour le résultant, de type Vandermonde pour l'évaluation et l'interpolation, de type Toeplitz et Hankel pour l'approximation de Padé et de Padé–Hermite, etc. Pour ces types de matrices clairement identifiés, on développe soit une algorithmique *ad hoc*, soit une théorie unifiée reposant sur le concept de « rang de déplacement ».

Schématiquement, l'algorithmique des matrices denses est la plus lente, de complexité située entre $O(n^2)$ et $O(n^3)$, en taille n. La complexité du calcul avec les matrices creuses est de l'ordre de $O(n^2)$, alors que celle des matrices structurées est en $O(n)$, à des facteurs logarithmiques près.

L'algorithmique des matrices denses constitue l'objet d'étude de ce chapitre. Les matrices creuses seront abordées brièvement au Chapitre 9 et les matrices structurées feront l'objet d'une étude plus approfondie au Chapitre 10.

Nous allons travailler avec un corps effectif noté $\mathbb{K}$ et avec l'algèbre des matrices carrées $\mathcal{M}_n(\mathbb{K})$ à coefficients dans $\mathbb{K}$. Signalons toutefois que la plupart des résultats de ce chapitre s'étendent au cas où le corps $\mathbb{K}$ est remplacé par un anneau effectif $\mathbb{A}$, et les matrices carrées par des matrices rectangulaires à coefficients dans $\mathbb{A}$.

Résultat principal

Les questions qui seront étudiées, ou simplement évoquées, dans ce chapitre sont celles de la complexité du calcul du produit matriciel dans $\mathcal{M}_n(\mathbb{K})$, de l'inverse, du déterminant, du polynôme caractéristique ou minimal, de la résolution de systèmes linéaires, ou encore de la mise sous diverses « formes canoniques » (échelon, LUP, compagnon par blocs, ...).

Pour ces questions, on dispose d'une première famille d'algorithmes « naïfs », reposant sur l'application directe des définitions ou des propriétés mathématiques. Si l'algorithme naïf de multiplication a une complexité raisonnable, cubique en la taille, dans d'autres cas l'emploi des algorithmes naïfs est vivement déconseillé. Par exemple, si $A = (a_{i,j})_{i,j=1}^{n}$ est une matrice de $\mathcal{M}_n(\mathbb{K})$, on n'utilisera pas la définition

$$\det(A) = \sum_{\sigma \in \mathcal{S}_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)}$$

pour le calcul de son déterminant ; en effet, cela mènerait à un algorithme de complexité $O(n \cdot n!)$, donc hyper-exponentielle en la taille de A. De même, la définition du rang d'une matrice comme la taille maximale d'un mineur non nul ne se prête pas directement, via une recherche exhaustive, à une algorithmisation efficace. Dans un registre légèrement différent, les formules de Cramer pour résoudre un système

linéaire ne sont pas d'une grande utilité pratique (en revanche, elles servent à borner les tailles des solutions, et s'avèrent souvent utiles dans les estimations de complexité).

Une seconde classe d'algorithmes, basés sur la *méthode d'élimination de Gauss*, permet de résoudre de manière raisonnablement efficace la plupart des problèmes évoqués plus haut. Appliquée à une matrice A de $\mathcal{M}_n(\mathbb{K})$, cette méthode fournit des algorithmes de complexité $O(n^3)$ pour le calcul du rang $\mathrm{rg}(A)$, du déterminant $\det(A)$, de l'inverse A^{-1} (si A est inversible), pour le calcul d'une forme échelonnée et d'une décomposition LUP de A, ainsi que pour la résolution d'un système linéaire de matrice A.

Le résultat principal de ce chapitre est que l'on peut faire mieux, et qu'il existe une constante $2 \leq \omega < 3$ (dépendante *a priori* du corps de base $\mathbb{K}$), qui contrôle la complexité du produit matriciel et de toutes les opérations d'algèbre linéaire.

Pour formaliser ce point, on associe à chacun des problèmes son *exposant*. Dans le cas du produit, il est défini comme suit.

Définition 8.1 — Exposant de la multiplication matricielle. On dit que le réel θ est un *exposant réalisable* pour la multiplication de matrices sur le corps $\mathbb{K}$ s'il existe un algorithme de complexité arithmétique $O(n^\theta)$ pour multiplier deux matrices arbitraires de $\mathcal{M}_n(\mathbb{K})$. On définit $\omega_{\mathrm{mul}} = \inf\{\theta \mid \theta \text{ est un exposant réalisable}\}$.

A priori, on devrait mettre en indice le corps $\mathbb{K}$ pour les exposants θ et ω_{mul}. Les résultats présentés dans la suite sont cependant indépendants de $\mathbb{K}$ [1]. L'algorithme naïf pour le produit est de complexité $O(n^3)$, donc $\theta = 3$ est un exposant réalisable et $\omega_{\mathrm{mul}} \leq 3$. D'un autre côté, ω_{mul} est forcément au moins égal à 2 : le nombre d'éléments d'une matrice est n^2 et il faut bien autant d'opérations pour l'écrire.

Ensuite, on peut définir de la même manière les exposants de tous les autres problèmes, de la forme ω_{inv} pour l'inversion, ω_{polcar} pour le polynôme caractéristique, $\omega_{\det}$ pour le déterminant, ω_{LUP} pour la décomposition LUP, etc. On a alors le résultat suivant.

Théorème 8.1 Les exposants ω_{mul}, ω_{inv}, ω_{polcar}, $\omega_{\det}$, ω_{LUP}, sont tous égaux, et inférieurs à 2,38 ; on note ω leur valeur commune. L'exposant correspondant à la résolution de systèmes linéaires est inférieur ou égal à ω.

Ce théorème contient deux (familles de) résultats :

— des preuves d'*équivalence* entre problèmes,

— des bornes supérieures sur leur complexité, c'est-à-dire des algorithmes.

Dans le présent chapitre, il est impossible de démontrer ce théorème dans son intégralité. Nous nous contenterons de prouver (et encore, sous des hypothèses simplificatrices) que le produit et l'inverse ont le même exposant, et que celui-ci est inférieur à $\log_2 7 < 2{,}81$.

Nous montrerons également que $\omega_{\det} \leq \omega_{\mathrm{polcar}} \leq \omega_{\mathrm{mul}} = \omega_{\mathrm{inv}}$ et laisserons en exercice (Exercice 8.13) la preuve de l'inégalité $\omega_{\mathrm{inv}} \leq \omega_{\det}$ qui permet de conclure que cette suite d'inégalités est en fait une suite d'égalités.

1. On peut montrer que la constante ω_{mul} ne dépend que de la caractéristique du corps de base $\mathbb{K}$ et il est conjecturé qu'elle est même entièrement indépendante du corps $\mathbb{K}$.

Applications

Les problèmes d'algèbre linéaire abordés ci-dessus sont très fréquemment rencontrés en pratique ; les résultats du Théorème 8.1 seront invoqués à plusieurs reprises dans cet ouvrage. Par exemple, l'algorithme de Beckermann–Labahn (Chapitre 7) pour le calcul d'approximants de Padé–Hermite repose ultimement sur des produits de matrices de polynômes. C'est aussi le cas de l'algorithme de Storjohann pour la résolution de systèmes linéaires à coefficients polynomiaux (Chapitre 11), et de l'algorithme de recherche de solutions séries d'équations algébriques (Chapitre 3) ou différentielles linéaires (Chapitre 13).

De nombreux algorithmes de résolution de systèmes polynomiaux reposent aussi sur de l'algèbre linéaire : les méthodes de calcul d'une base de Gröbner (Chapitre 23) peuvent se ramener à des systèmes linéaires (éventuellement creux) en très grande taille ; les résultats de complexité que nous donnerons pour les calculs de bases de Gröbner (Chapitre 26) reposent sur le calcul rapide de forme échelon ; l'algorithme de « résolution géométrique » (Chapitre 28) utilise une itération de Newton faisant intervenir des calculs d'inverses de matrices de séries formelles. Dans un contexte différent, les algorithmes de factorisation des polynômes (Partie IV), et ceux pour les séries différentiellement finies (Chapitre 14) et les algorithmes de sommation et d'intégration (Chapitres 29 et 32) font intervenir comme sous-problème la résolution d'un système linéaire. L'algèbre linéaire est au cœur de nombreux autres problèmes algorithmiques qui ne seront pas abordés dans cet ouvrage ; c'est le cas de la factorisation des entiers, ou encore du logarithme discret en cryptanalyse. Mentionnons enfin qu'une large partie des ordinateurs du monde passent leurs cycles à faire des calculs d'algèbre linéaire, que ce soit pour faire des calculs d'optimisation combinatoire (programmation linéaire par l'algorithme du simplexe), de la simulation numérique (méthode des éléments finis), ou de la recherche de pages web (le système de classement PageRank utilisé par le moteur de recherche Google repose sur la recherche d'un vecteur propre d'une gigantesque matrice creuse).

8.2 Multiplication de matrices

Tout comme le produit d'entiers et de polynômes, la multiplication matricielle est une opération fondamentale en calcul formel, dont l'étude de complexité s'avère être un problème hautement non trivial.

L'algorithme naïf pour effectuer le produit d'une matrice de taille $m \times n$ par un vecteur de taille n utilise $O(mn)$ opérations arithmétiques : mn multiplications et $m(n-1)$ additions. Un résultat de Winograd affirme qu'en général on ne peut pas faire mieux : par exemple, pour les matrices réelles dont les coefficients sont algébriquement indépendants sur $\mathbb{Q}$, l'algorithme naïf *est optimal*. Puisque le produit matriciel peut s'interpréter comme une succession de produits matrice-vecteur, une question naturelle est de savoir si la multiplication *naïve* de deux matrices est aussi quasi-optimale.

La réponse à cette question, et les résultats les plus importants de cette section, classés par ordre chronologique de découverte, sont regroupés dans le théorème suivant.

Théorème 8.2 — La multiplication matricielle naïve n'est pas optimale. Soit $\mathbb{K}$ un corps. On peut multiplier deux matrices de $\mathcal{M}_n(\mathbb{K})$ en

a. $O(n^3)$ opérations dans $\mathbb{K}$, par l'algorithme naïf ;
b. $n^2\lceil\frac{n}{2}\rceil + 2n\lfloor\frac{n}{2}\rfloor \simeq \frac{1}{2}n^3 + n^2$ multiplications dans $\mathbb{K}$;
c. $n^2\lceil\frac{n}{2}\rceil + (2n-1)\lfloor\frac{n}{2}\rfloor \simeq \frac{1}{2}n^3 + n^2 - \frac{n}{2}$ multiplications dans $\mathbb{K}$ si la caractéristique de $\mathbb{K}$ est différente de 2 et si la division par 2 est gratuite ;
d. $O(n^{\log_2 7}) \simeq O(n^{2,81})$ opérations dans $\mathbb{K}$.

Ces résultats restent valables sur un anneau commutatif $\mathbb{A}$; pour (a) et (d) l'hypothèse de commutativité peut même être supprimée. La partie (b) est due à Winograd et fut historiquement la première amélioration de l'algorithme naïf. Elle réduit de moitié le nombre de multiplications, en augmentant en revanche le nombre d'additions. Cela constitue déjà un progrès pour une large classe d'anneaux $\mathbb{A}$ dans lesquels la multiplication est plus coûteuse que l'addition [2]. Par exemple, la technique du « scindage binaire » qui sera utilisée au Chapitre 15 pour le calcul d'un terme d'une récurrence, ou encore pour le calcul du développement décimal de e, requiert de multiplier des matrices contenant de gros nombres entiers.

L'amélioration (c) a été obtenue par Waksman, et implique qu'on peut effectuer le produit de deux matrices 2×2 en 7 multiplications. L'inconvénient commun des algorithmes de Winograd et de Waksman est l'utilisation de la commutativité de $\mathbb{K}$, qui devient un obstacle à une application récursive, ne permettant pas d'améliorer (la borne supérieure 3 sur) l'exposant ω_{mul}. Beaucoup pensaient que tout résultat de type (b) et (c) serait optimal, au sens que $\frac{1}{2}n^3$ multiplications dans $\mathbb{K}$ seraient nécessaires pour le produit dans $\mathcal{M}_n(\mathbb{K})$. Ceci fut contredit par la découverte par Strassen de (d) ; ce résultat découle d'un fait d'une simplicité déconcertante, mais d'une portée considérable : deux matrices 2×2 peuvent être multipliées en 7 multiplications même si $\mathbb{K}$ n'est pas commutatif.

La non-optimalité de la multiplication matricielle naïve justifie l'introduction de la définition suivante.

Définition 8.2 — Fonction de multiplication matricielle. On dit que l'application $\mathsf{MM} : \mathbb{N} \to \mathbb{N}$ est une *fonction de multiplication matricielle* (pour un corps $\mathbb{K}$) si :

— on peut multiplier deux matrices arbitraires de $\mathcal{M}_n(\mathbb{K})$ en au plus $\mathsf{MM}(n)$ opérations dans $\mathbb{K}$;
— MM est croissante et vérifie l'inégalité $\mathsf{MM}(n/2) \leq \mathsf{MM}(n)/4$ pour $n \in \mathbb{N}$.

Ainsi, le Théorème 8.2 montre que les fonctions $n \mapsto n^3$ ou $n \mapsto n^{\log_2 7}$ sont (à des constantes près) des fonctions de multiplication matricielles. Comme dans le cas de la fonction de multiplication polynomiale M introduite au Chapitre 2, l'intérêt de cette notion est qu'elle permet d'énoncer des résultats de complexité indépendants du choix de l'algorithme utilisé pour multiplier les matrices.

La seconde condition de la Définition 8.2 établit la cohérence avec la Définition 8.1 : si $\mathsf{MM}(n) = c \cdot n^\theta$, pour une certaine constante $c > 0$, alors θ doit être supérieur ou égal

2. C'est par exemple le cas de $\mathbb{Z}$ et de $\mathbb{Z}[X]$, mais pas de $\mathbb{Q}$ ou de $\mathbb{Q}(X)$.

à 2. Cette condition permettra de montrer que dans des algorithmes de type « diviser pour régner » pour les matrices, le coût total est essentiellement celui du dernier appel récursif.

Le reste de cette section est dédié à la preuve du Théorème 8.2.

Multiplication naïve

L'algorithme « cubique » de multiplication prend en entrée deux matrices $A = (a_{i,j})_{i,j=1}^n$ et $X = (x_{i,j})_{i,j=1}^n$ de $\mathcal{M}_n(\mathbb{K})$, et calcule leur produit $R = AX$ en utilisant la formule

$$r_{i,k} = \sum_{j=1}^{n} a_{i,j} x_{j,k}.$$

Le coût nécessaire pour obtenir un élément de R est donc de n multiplications et $n-1$ additions, de sorte que la complexité totale est en $O(n^3)$.

Algorithme de Winograd

Il est possible de diviser essentiellement par deux le nombre de multiplications scalaires suffisantes pour le calcul du produit AX. Ceci montre au passage que la multiplication matricielle naïve n'est pas optimale. Supposons pour simplifier que n est pair, $n = 2k$. L'algorithme de Winograd repose sur l'identité suivante, héritière de celle de Karatsuba : si ℓ est un vecteur ligne $(a_1 \cdots a_n)$ et c est un vecteur colonne ${}^t(x_1 \cdots x_n)$, alors le produit scalaire $(\ell \mid c) = \sum_i a_i x_i$ s'écrit

$$(\ell|c) = (a_1 + x_2)(a_2 + x_1) + \cdots + (a_{2k-1} + x_{2k})(a_{2k} + x_{2k-1}) - \sigma(\ell) - \sigma(c), \qquad (8.1)$$

où $\sigma(\ell) = a_1 a_2 + \cdots + a_{2k-1} a_{2k}$ et $\sigma(c) = x_1 x_2 + \cdots + x_{2k-1} x_{2k}$. Notons que $\sigma(\ell)$ se calcule en $k = n/2$ multiplications et $k - 1 = n/2 - 1$ additions.

Si maintenant $\ell_1, \ldots, \ell_n$ sont les lignes de A et $c_1, \ldots, c_n$ sont les colonnes de X, l'élément (i, j) du produit AX vaut $(\ell_i|c_j)$. L'idée est alors de précalculer les quantités $\sigma(\ell_i)$ et $\sigma(c_i)$ pour $1 \le i \le n$; ce précalcul demande $n(\frac{n}{2} + \frac{n}{2}) = n^2$ multiplications et $n(\frac{n}{2} - 1 + \frac{n}{2} - 1) = n^2 - 2n$ additions. Une fois déterminés les $\sigma(\ell_i)$ et $\sigma(c_i)$, on peut calculer l'ensemble des éléments $r_{i,j}$ de $R = AX$ grâce à la formule (8.1) en $n^2 \cdot \frac{n}{2} = \frac{1}{2}n^3$ multiplications et $n^2 \cdot (n + (\frac{n}{2} - 1) + 2) = \frac{3}{2}n^3 + n^2$ additions. Au total, le coût de la multiplication de l'algorithme de Winograd est de $\frac{1}{2}n^3 + n^2$ multiplications et $\frac{3}{2}n^3 + 2n^2 - 2n$ additions. Ainsi, on a essentiellement transformé $n^3/2$ multiplications en additions, ce qui est appréciable.

Algorithme de Waksman

Pour effectuer le produit de matrices 2×2

$$R = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} x & y \\ z & t \end{pmatrix},$$

l'algorithme de Winograd ci-dessus utilise 8 multiplications, en écrivant

$$R = \begin{pmatrix} (a+z)(b+x) - ab - zx & (a+t)(b+y) - ab - ty \\ (c+z)(d+x) - cd - zx & (c+t)(d+y) - cd - ty \end{pmatrix}.$$

Il n'apporte donc aucun gain par rapport à l'algorithme naïf dans cette taille.

L'algorithme suivant, dû à Waksman, peut être vu comme une amélioration de l'algorithme de Winograd. Il effectue le produit de matrices 2×2 en 7 opérations, dès lors que dans le corps de base $\mathbb{K}$ l'élément 2 est inversible et que la division par 2 y est gratuite. L'idée est d'écrire la matrice R comme

$$\mathrm{R} = \frac{1}{2}\begin{pmatrix} \alpha_1 - \alpha_2 & \beta_1 - \beta_2 \\ \gamma_1 - \gamma_2 & \delta_1 - \delta_2 \end{pmatrix},$$

où

$$\begin{aligned} \alpha_1 &= (a+z)(b+x), & \alpha_2 &= (a-z)(b-x), & \beta_1 &= (a+t)(b+y), & \beta_2 &= (a-t)(b-y), \\ \gamma_1 &= (c+z)(d+x), & \gamma_2 &= (c-z)(d-x), & \delta_1 &= (c+t)(d+y), & \delta_2 &= (c-t)(d-y), \end{aligned}$$

et de faire observer que, grâce à l'identité

$$(\gamma_1 + \gamma_2) + (\beta_1 + \beta_2) = (\alpha_1 + \alpha_2) + (\delta_1 + \delta_2) = 2(ab + cd + xz + ty),$$

l'élément δ_2 est une combinaison linéaire des sept autres produits $\alpha_1, \alpha_2, \ldots, \delta_1$, et donc n'a pas besoin d'être obtenu lui aussi par un produit.

L'algorithme qui s'en déduit, tout comme celui de Winograd, ne peut être utilisé de manière récursive : la preuve des formules sous-jacentes repose sur le fait que les éléments du corps de base $\mathbb{K}$ commutent ($bz = zb, bt = tb, \ldots$). Lors des appels récursifs, les objets à multiplier seraient eux-mêmes des matrices, pour lesquelles la loi de commutation n'est plus vérifiée. On dit donc que les algorithmes de Winograd et de Waksman sont des algorithmes *commutatifs*.

Exercice 8.1 Finir la preuve des parties (b) et (c) du Théorème 8.2, en généralisant les arguments précédents à des matrices de taille n, où $n \in \mathbb{N}$ n'est pas nécessairement pair. ■

Algorithme de Strassen

L'algorithme de Strassen fut le premier à descendre en-dessous de $O(n^3)$. Tout comme l'algorithme de Karatsuba pour les polynômes, il repose sur le gain d'une multiplication pour les matrices 2×2, qui devient un gain dans l'exposant lors d'une application récursive. La non-commutativité du produit matriciel se traduit en revanche par une contrainte supplémentaire : afin de se prêter à un emploi récursif, le nouvel algorithme doit, à la différence de l'algorithme de Waksman, satisfaire à une contrainte de non-commutativité.

Soient donc A et X des matrices de taille 2 à multiplier par un algorithme non commutatif. Intuitivement, cela se traduit par le fait que lors des multiplications utilisées par un tel algorithme, les éléments a, b, c, d de A doivent rester à gauche, et les éléments x, y, z, t de X à droite (cela n'est pas le cas pour les algorithmes de Winograd et de Waksman, qui *mélangent* les éléments de A et X).

L'algorithme naïf est bien non commutatif, mais demande d'effectuer 8 multiplications. L'algorithme de Strassen que nous allons présenter revient à

— calculer des combinaisons linéaires des éléments de A et des éléments de X,

Entrée Deux matrices $A, X \in \mathcal{M}_n(\mathbb{K})$, avec $n = 2^k$.
Sortie Le produit AX.

1. Si $n = 1$, renvoyer AX.
2. Décomposer $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ et $X = \begin{pmatrix} x & y \\ z & t \end{pmatrix}$, avec a, b, c, d et x, y, z, t dans $\mathcal{M}_{n/2}(\mathbb{K})$.
3. Calculer récursivement les produits

$$\begin{aligned} q_1 &= a(x+z), & q_2 &= d(y+t), \\ q_3 &= (d-a)(z-y), & q_4 &= (b-d)(z+t) \\ q_5 &= (b-a)z, & q_6 &= (c-a)(x+y), & q_7 &= (c-d)y. \end{aligned}$$

4. Calculer les sommes

$$\begin{aligned} r_{1,1} &= q_1 + q_5, & r_{1,2} &= q_2 + q_3 + q_4 - q_5, \\ r_{2,1} &= q_1 + q_3 + q_6 - q_7, & r_{2,2} &= q_2 + q_7. \end{aligned}$$

5. Renvoyer $\begin{pmatrix} r_{1,1} & r_{1,2} \\ r_{2,1} & r_{2,2} \end{pmatrix}$.

Algorithme 8.1 – Algorithme de Strassen pour multiplier deux matrices.

— effectuer **7** produits de ces combinaisons linéaires,
— recombiner les résultats pour obtenir les quatre éléments du produit AX.

Explicitement, les formules à appliquer sont données dans les étapes (3) et (4) de l'Algorithme 8.1. Effectuons maintenant le pas récursif. Quitte à rajouter des colonnes et des lignes nulles, on peut supposer que les matrices A et X sont de taille $n = 2^k$, avec $k \in \mathbb{N}$. On procède à une découpe en blocs de A et de X :

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \qquad X = \begin{pmatrix} x & y \\ z & t \end{pmatrix}.$$

Dans cette écriture a, b, c, d et x, y, z, t sont donc des matrices carrées de taille $n/2$.

Le point-clé est le suivant : grâce à la non-commutativité, les formules données dans l'Algorithme 8.1 pour le cas 2×2 s'appliquent toujours si a, b, c, d et x, y, z, t sont des matrices. Ainsi, elles permettent de ramener le produit en taille n à 7 produits en taille $n/2$, plus un certain nombre C (ici C = 18) d'additions en taille $n/2$, utilisées pour fabriquer les combinaisons linéaires des blocs de A et de X, et celles des produits q_i.

La complexité $\mathsf{S}(n)$ de l'algorithme de Strassen satisfait donc à la récurrence

$$\mathsf{S}(n) \leq 7\,\mathsf{S}\left(\frac{n}{2}\right) + \mathrm{C}\left(\frac{n}{2}\right)^2.$$

En invoquant le lemme « diviser pour régner » avec les choix $m = 7, p = s = 2$, $\kappa = 1$ et $q = 4$, on déduit l'inégalité

$$\mathsf{S}(n) \leq \left(1 + \frac{\mathrm{C}}{3}\right) n^{\log_2 7},$$

qui prouve la partie (d) du Théorème 8.2.

L'idée de base de l'algorithme de Strassen est tout à fait générale : améliorer la multiplication en petite taille permet d'améliorer l'exposant.

Exercice 8.2 Supposons que, pour un certain entier r fixé, on sache multiplier deux matrices de taille r à coefficients dans un corps $\mathbb{K}$ en r^θ multiplications non commutatives dans $\mathbb{K}$ et un nombre arbitraire d'additions dans $\mathbb{K}$. Montrer que θ est un exposant réalisable pour la multiplication des matrices à coefficients dans $\mathbb{K}$. ∎

En pratique, il peut être intéressant de ne pas descendre récursivement jusqu'à la multiplication des matrices 1×1, mais d'arrêter la récursion à une taille $s > 1$ et d'utiliser l'algorithme naïf (ou tout autre algorithme, même commutatif !) en cette taille. Cela permet d'optimiser la constante du $O(\cdot)$.

Exercice 8.3 Soit n une puissance de 2. Établir un algorithme hybride de multiplication matricielle en taille $n \times n$, qui fait appel à l'algorithme de Strassen pour $n \geq 2^d$ et à l'algorithme naïf pour $n < 2^d$. Montrer que la complexité $\mathsf{MM}(n)$ de cet algorithme vérifie $\mathsf{MM}(n) \leq \mu(d) n^{\log_2 7}$ pour tout $n \geq 2^d$, où $\mu(d)$ est une fonction qui dépend uniquement de d. Trouver la valeur de d qui minimise $\mu(d)$. ∎

Lorsque la taille n des matrices à multiplier n'est pas une puissance de 2, on peut rajouter artificiellement des lignes et des colonnes aux deux matrices de façon à se ramener au cas où n est une puissance de 2.

Exercice 8.4 Écrire une version récursive de l'algorithme de Strassen valable en taille arbitraire n (non nécessairement une puissance de 2), et qui procède en rajoutant une ligne et une colonne lorsque n est impair. Montrer que la complexité de cet algorithme vérifie $\mathsf{MM}(n) \leq C(n) n^{\log_2 7}$ et comparer $C(n)$ à ce que l'on obtient en remplaçant n par la puissance de 2 supérieure à n. ∎

L'algorithme de Strassen s'applique aussi au cas où le corps de base $\mathbb{K}$ est remplacé par un anneau $\mathbb{A}$. Lorsque cet anneau $\mathbb{A}$ est une extension de $\mathbb{K}$, la question est d'estimer sa complexité en termes de nombre d'opérations dans $\mathbb{K}$.

Exercice 8.5 Soit $\mathbb{K}$ un corps et soient d et n deux entiers naturels strictement positifs. Estimer la complexité de l'algorithme de multiplication de Strassen appliqué à deux matrices carrées de taille n, dont les éléments sont des polynômes de degré au plus d de $\mathbb{K}[X]$. ∎

Plus généralement, tout algorithme de multiplication dans $\mathcal{M}_n(\mathbb{K})$ utilisant $\mathsf{MM}(n) = O(n^\theta)$ opérations dans $\mathbb{K}$ fournit par application à $\mathbb{A} = \mathbb{K}[X]/(X^{2d+1})$ un algorithme de multiplication de matrices polynomiales de degré d de complexité $\mathsf{MM}(n,d) = O(n^\theta \mathsf{M}(d))$ opérations de $\mathbb{K}$. Un algorithme de meilleure complexité permettant d'obtenir $\mathsf{MM}(n,d) = O(n^\theta d + n^2 \mathsf{M}(d))$ sera présenté au Chapitre 5.

Interprétation des formules de Strassen

Il n'est pas immédiat de donner une intuition aux formules de Strassen (à la différence de celle de Karatsuba, qui repose de manière cachée sur l'évaluation de polynômes en $0, 1, +\infty$). Nous allons présenter un raisonnement qui permet de motiver

les formules de Strassen et de les généraliser. L'explication appartient à Fiduccia ; elle est postérieure à la découverte de l'algorithme [3].

Le point de départ est l'observation que la recherche d'un algorithme non commutatif pour effectuer le produit de matrices R = AX avec

$$\mathrm{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \qquad \text{et} \qquad \mathrm{X} = \begin{pmatrix} x & y \\ z & t \end{pmatrix}$$

revient à donner un algorithme pour la multiplication matrice-vecteur Mv, où

$$\mathrm{M} = \begin{pmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{pmatrix} \qquad \text{et} \qquad v = \begin{pmatrix} x \\ z \\ y \\ t \end{pmatrix}.$$

La suite de l'argument repose sur la remarque suivante : toute matrice (de taille arbitraire) de la forme

$$\begin{pmatrix} \alpha & \alpha \\ \varepsilon\alpha & \varepsilon\alpha \end{pmatrix}, \begin{pmatrix} \alpha & -\alpha \\ \varepsilon\alpha & -\varepsilon\alpha \end{pmatrix} \text{ou} \begin{pmatrix} \alpha & \varepsilon\alpha \\ -\alpha & -\varepsilon\alpha \end{pmatrix},$$

où $\varepsilon \in \{0, 1\}$ et $\alpha \in \mathbb{K}$, dont tous les éléments, sauf les quatre marqués explicitement, sont nuls, peut être multipliée par un vecteur en une seule multiplication dans $\mathbb{K}$. On appelle une telle matrice *élémentaire*.

L'idée est alors de décomposer la matrice M en somme de 7 matrices élémentaires. On commence par soustraire à M les deux matrices élémentaires

$$\mathrm{E}_1 = \begin{pmatrix} a & a & & \\ a & a & & \\ & & & \\ & & & \end{pmatrix} \qquad \text{et} \qquad \mathrm{E}_2 = \begin{pmatrix} & & & \\ & & & \\ & & d & d \\ & & d & d \end{pmatrix},$$

correspondant aux éléments diagonaux de A. La matrice ainsi obtenue est

$$\mathrm{M}_1 = \begin{pmatrix} & & b-a & \\ c-a & d-a & & \\ & & & a-d \quad b-d \\ & & & c-d \end{pmatrix}.$$

3. Strassen lui-même ne se souvient pas exactement de l'origine de ses formules. Selon Landsberg, « *Strassen was attempting to prove, by process of elimination, that such an algorithm did not exist when he arrived at it* ». En tout cas, Strassen affirme « *First I had realized that an estimate tensor rank* < 8 *for two by two matrix multiplication would give an asymptotically faster algorithm. Then I worked over* $\mathbb{Z}/2\mathbb{Z}$ *— as far as I remember — to simplify matters* ». Autrement dit, il y a dû y avoir recherche « à la main ».

Par construction, elle admet la propriété que l'élément $(1,1)$ du deuxième bloc est l'opposé de l'élément $(2,2)$ du premier. En soustrayant à M_1 la matrice élémentaire

$$\mathrm{E}_3 = \begin{pmatrix} d-a & a-d \\ d-a & a-d \end{pmatrix}$$

on obtient la matrice

$$\mathrm{M}_2 = \begin{pmatrix} & b-a & & \\ c-a & & d-a & \\ & a-d & & b-d \\ & & c-d & \end{pmatrix},$$

qui se décompose comme $\mathrm{M}_3 + \mathrm{M}_4$, avec

$$\mathrm{M}_3 = \begin{pmatrix} b-a & \\ a-d & b-d \end{pmatrix} \quad \text{et} \quad \mathrm{M}_4 = \begin{pmatrix} c-a & d-a \\ & c-d \end{pmatrix}.$$

Or, puisque $a-d = (b-d)-(b-a)$ et $d-a = (c-a)-(c-d)$, chacune des matrices M_3 et M_4 s'écrit comme somme de 2 matrices élémentaires

$$\mathrm{M}_3 = \mathrm{E}_4 + \mathrm{E}_5, \quad \text{où} \quad \mathrm{E}_4 = \begin{pmatrix} b-a & \\ a-b & \end{pmatrix}, \quad \mathrm{E}_5 = \begin{pmatrix} b-d & b-d \\ & \end{pmatrix},$$

et

$$\mathrm{M}_4 = \mathrm{E}_6 + \mathrm{E}_7, \quad \text{où} \quad \mathrm{E}_6 = \begin{pmatrix} c-a & c-a \\ & \end{pmatrix}, \quad \mathrm{E}_7 = \begin{pmatrix} d-c & \\ c-d & \end{pmatrix}.$$

En revenant à l'égalité ${}^t\begin{pmatrix} r_{1,1} & r_{2,1} & r_{1,2} & r_{2,2} \end{pmatrix} = \left(\sum_{i=1}^{7} \mathrm{E}_i\right) {}^t v$ traduisant $\mathrm{R} = \mathrm{AX}$, on obtient finalement les formules données dans les étapes (3) et (4) de l'Algorithme 8.1.

Exercice 8.6 Généraliser l'algorithme de Strassen aux matrices de taille arbitraire. Plus précisément, étendre la preuve des formules de Strassen pour le produit matriciel en taille 2×2 afin de montrer que le produit de deux matrices carrées de taille n peut s'effectuer en $n^3 - n(n-1)/2$ multiplications scalaires non commutatives.

Est-ce que cela permet d'améliorer la borne de Strassen $\log_2 7$ sur l'exposant de l'algèbre linéaire ? ■

Peut-on faire mieux que 2,81 ?

Strassen ayant prouvé que l'exposant ω_{mul} du produit de matrices était sous-cubique, la question naturelle est devenue de savoir quelle est sa véritable valeur.

Pour ce faire, depuis les années 1970, de nombreux outils ont été développés, les uns plus sophistiqués que les autres, et l'on fit preuve de beaucoup d'ingéniosité. Des notions nouvelles sont apparues, comme celles de *complexité bilinéaire*, de *rang tensoriel* et d'*algorithme approché*, utilisées de manière intensive notamment par Bini, Pan, Schönhage, Strassen, Winograd et d'autres. Ces notions ne seront pas abordées dans cet ouvrage.

Le meilleur algorithme actuel, datant de 2014 et dû à François Le Gall, est de complexité $O(n^{2,3728639})$. Il est cependant conjecturé que $\omega_{\text{mul}} = 2$, mais pour l'instant la preuve (ou la réfutation) de cette conjecture semble hors de portée.

En pratique

Les algorithmes rapides pour l'algèbre linéaire ont longtemps eu mauvaise presse ; concrètement, l'algorithme de Strassen, et, de manière plus marginale, un algorithme d'exposant 2,77 dû à Pan et repris par Kaporin sont les seuls algorithmes « rapides » (avec un exposant inférieur à 3) qui ont été implantés avec succès.

Dans une bonne implantation, disons sur un corps fini défini modulo un entier de taille « raisonnable », les algorithmes de Winograd et de Waksman sont immédiatement rentables. L'algorithme de Strassen devient ensuite meilleur pour des tailles de l'ordre de quelques dizaines (64 est une borne raisonnable).

L'immense majorité des autres algorithmes connus reposent sur des techniques très complexes, qui se traduisent par la présence d'énormes constantes (et de facteurs logarithmiques) dans leurs estimations de complexité. En conséquence, dans leurs versions actuelles, il ne peuvent pas être rentables pour des tailles inférieures à des millions ou des milliards …

8.3 Autres problèmes d'algèbre linéaire

Il est important de savoir multiplier les matrices, mais il est encore plus utile de les inverser, de calculer leur polynôme caractéristique, etc.

Élimination de Gauss

Rappelons que l'algorithme classique d'élimination de Gauss permet de résoudre la plupart des problèmes classiques en algèbre linéaire de manière bien plus efficace que l'algorithmique naïve.

Définition 8.3 — Forme échelonnée. Une matrice est *échelonnée en ligne* lorsque
1. les lignes nulles sont toutes en dessous des lignes non nulles,
2. le premier coefficient non nul (appelé *pivot*) de chaque ligne non nulle est strictement à droite du premier coefficient non nul de la ligne précédente.

Une forme *échelonnée en ligne* d'une matrice A est une matrice échelonnée en ligne B équivalente à gauche à A (c'est-à-dire telle qu'il existe une matrice carrée inversible P avec A = PB).

L'algorithme de Gauss (sans pivot sur les colonnes) calcule, en n'utilisant que des opérations élémentaires sur les lignes (qui correspondent à des multiplications à gauche par des matrices inversibles), une forme échelonnée en ligne de A, sur laquelle le rang et le déterminant de A se lisent directement.

En effet, les lignes non nulles d'une forme échelonnée B forment une base de l'espace vectoriel engendré par les lignes de la matrice de départ A. Le nombre de lignes non nulles de B est égal au rang de A. Si A est carrée, son déterminant est égal au produit des éléments diagonaux de B.

Une variante de l'algorithme de Gauss, dite de *Gauss–Jordan*, produit même une *forme échelonnée réduite* de A : il s'agit d'une forme échelonnée de A, dont les pivots valent 1, les autres coefficients dans les colonnes des pivots étant nuls. Appliquée à la matrice augmentée $\tilde{A} = (A|I_n)$ obtenue par concaténation de la matrice A et de la matrice identité I_n, cette variante est utilisée dans le calcul de l'inverse A^{-1}. En effet, l'algorithme de Gauss–Jordan transforme la matrice $\tilde{A}$ en une matrice équivalente dont le bloc gauche est l'identité, c'est-à-dire qu'il remplace $\tilde{A}$ par la matrice $(I_n|A^{-1})$. Le même algorithme permet de résoudre le système $Ax = b$, où $b \in \mathbb{K}^n$, en bordant la matrice A par le vecteur b.

Une autre variante de l'algorithme d'élimination de Gauss permet de calculer une *décomposition LUP* qui, à son tour, permet de résoudre des systèmes linéaires, d'inverser des matrices, de calculer des déterminants, etc.

Définition 8.4 — Matrices L, U, P et décomposition LUP. Une matrice est dite *triangulaire supérieure*, resp. *triangulaire inférieure*, si les éléments situés en dessous (resp. au dessus) de la diagonale principale sont nuls. Une *matrice de permutation* est une matrice carrée P dont les coefficients valent 0 ou 1, et dont chaque ligne et colonne ne contient qu'un seul élément non nul. Une *décomposition LU*, resp. *LUP*, d'une matrice A est une factorisation de la forme A = LU, resp. A = LUP, avec L triangulaire inférieure, U triangulaire supérieure et P une matrice de permutation.

En résumé, nous allons admettre l'énoncé suivant.

Théorème 8.3 — Élimination gaussienne. Pour toute matrice $A \in \mathcal{M}_n(\mathbb{K})$, il est possible de calculer en $O(n^3)$ opérations dans $\mathbb{K}$:

1. le rang $rg(A)$, le déterminant $\det(A)$, et l'inverse A^{-1} si A est inversible ;
2. une base (affine) de solutions de $Ax = b$, pour tout b dans $\mathbb{K}^n$;
3. une décomposition LUP, et une forme échelonnée réduite de A.

Résultat principal

Théorème 8.4 — L'élimination gaussienne n'est pas optimale. Soit $\mathbb{K}$ un corps et soit θ un exposant réalisable pour la multiplication des matrices à coefficients dans $\mathbb{K}$. Pour toute matrice A de $\mathcal{M}_n(\mathbb{K})$, il est possible de calculer :

a. le déterminant $\det(A)$;
b. l'inverse A^{-1} (si A est inversible) ;
c. la solution $x \in \mathbb{K}^n$ du système $Ax = b$, pour tout $b \in \mathbb{K}^n$, si A est inversible ;

d. le polynôme caractéristique de A ;
e. une décomposition LUP de A ;
f. le rang $rg(A)$ et une forme échelonnée réduite de A ;
g. une base du noyau de A ;

en $\tilde{O}(n^\theta)$ opérations dans $\mathbb{K}$.

La preuve des parties (e), (f) et (g) dépasse le cadre de cet ouvrage. En termes d'exposants, les assertions (a) et (d) impliquent les inégalités $\omega_{inv} \leq \omega_{mul}$ et $\omega_{polcar} \leq \omega_{mul}$. L'inégalité $\omega_{det} \leq \omega_{polcar}$ est évidente, puisque le terme constant du polynôme caractéristique de A est égal à $(-1)^n \det(A)$.

Dans la suite, nous allons prouver (a)–(d), sous des hypothèses simplificatrices. Nous montrerons également que $\omega_{mul} \leq \omega_{inv}$ et laisserons en exercice (Exercice 8.13) la preuve de $\omega_{inv} \leq \omega_{det}$. Utilisées conjointement, ces inégalités permettent de prouver le Théorème 8.1.

La multiplication n'est pas plus difficile que l'inversion

Soient A et B des matrices $n \times n$. On souhaite calculer $C = AB$. Pour cela, on pose

$$D = \begin{pmatrix} I_n & A & 0 \\ 0 & I_n & B \\ 0 & 0 & I_n \end{pmatrix}.$$

On a alors l'identité remarquable

$$D^{-1} = \begin{pmatrix} I_n & -A & AB \\ 0 & I_n & -B \\ 0 & 0 & I_n \end{pmatrix},$$

qui permet donc de ramener le produit en taille n à l'inversion en taille $3n$. Cela prouve notre affirmation et l'inégalité

$$\omega_{mul} \leq \omega_{inv}.$$

Exercice 8.7 Soit $\mathbb{K}$ un corps et soit $T(n)$ la complexité du produit de deux matrices triangulaires inférieures de taille $n \times n$ et à coefficients dans $\mathbb{K}$. Montrer qu'il est possible de multiplier deux matrices arbitraires de $\mathcal{M}_n(\mathbb{K})$ en $O(T(n))$ opérations dans $\mathbb{K}$. ■

L'inversion, le calcul de déterminant et la résolution de système ne sont pas plus difficiles que la multiplication.

Nous allons montrer qu'il est possible d'inverser des matrices $n \times n$ par un algorithme de type « diviser pour régner » en $\tilde{O}(n^\theta)$ opérations arithmétiques, quel que soit l'exposant réalisable θ.

Inverse

L'algorithme d'inversion matricielle présenté ici, dû à Strassen, est un algorithme récursif, qui peut s'interpréter comme un « pivot de Gauss par blocs ». Cet algorithme nécessite d'inverser certaines sous-matrices de A ; afin de garantir sa correction, nous

Entrée Une matrice « générique » $A \in \mathcal{M}_n(\mathbb{K})$, avec $n = 2^k$.
Sortie Son inverse A^{-1}.

1. Si $n = 1$, renvoyer A^{-1}.
2. Décomposer $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, avec a, b, c, d dans $\mathcal{M}_{n/2}(\mathbb{K})$.
3. Calculer $e := a^{-1}$ récursivement.
4. Calculer $Z := d - ceb$.
5. Calculer $t := Z^{-1}$ récursivement.
6. Calculer $y := -ebt$, $z := -tce$, $x := e + ebtce$.
7. Renvoyer $\begin{pmatrix} x & y \\ z & t \end{pmatrix}$.

Algorithme 8.2 – Algorithme de Strassen pour inverser une matrice.

allons faire l'hypothèse simplificatrice que *toutes ces matrices sont inversibles*. Le traitement du cas général (A matrice arbitraire) est plus délicat, et passe par le calcul efficace, toujours de type « diviser pour régner », d'une décomposition LUP de A.

Le point de départ est l'identité suivante (non commutative !), dans laquelle on suppose $n = 2$ et $a, Z \in \mathbb{K} \setminus \{0\}$:

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ ca^{-1} & 1 \end{pmatrix} \times \begin{pmatrix} a & 0 \\ 0 & Z \end{pmatrix} \times \begin{pmatrix} 1 & a^{-1}b \\ 0 & 1 \end{pmatrix}, \tag{8.2}$$

et où $Z = d - ca^{-1}b$ est le *complément de Schur* de a dans A.

Cette identité se déduit aisément grâce à la méthode du pivot de Gauss appliquée à A et permet d'obtenir la factorisation suivante

$$\begin{aligned} \begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} &= \begin{pmatrix} 1 & -a^{-1}b \\ 0 & 1 \end{pmatrix} \times \begin{pmatrix} a^{-1} & 0 \\ 0 & Z^{-1} \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ -ca^{-1} & 1 \end{pmatrix} \\ &= \begin{pmatrix} a^{-1} + a^{-1}bZ^{-1}ca^{-1} & -a^{-1}bZ^{-1} \\ -Z^{-1}ca^{-1} & Z^{-1} \end{pmatrix}. \end{aligned} \tag{8.3}$$

Le point important est que l'identité (8.3) est « non commutative » : elle reste valable dans tout anneau non commutatif, pourvu que les éléments a et $Z = d - ca^{-1}b$ soient inversibles. Elle permet donc une application récursive. L'algorithme 8.2 résume la méthode d'inversion. Dans chaque appel récursif l'algorithme effectue deux inversions, ainsi qu'un certain nombre de produits. Plus précisemment, avec les notations de l'Algorithme 8.2, les 6 produits ce, bt, $ceb = (ce)b$, $ebt = e(bt)$, $(ebt)(ce)$ et $t(ce)$ sont suffisants. Cette analyse mène au résultat suivant.

Théorème 8.5 Si toutes les sous-matrices rencontrées au cours de l'Algorithme 8.2 sont inversibles, le coût de l'inversion de A est de $3\mathsf{MM}(n) + O(n^2)$.

Démonstration. L'algorithme ci-dessus montre que la complexité $\mathsf{I}(n)$ de l'inversion

satisfait à la récurrence

$$\mathsf{I}(n) \leq 2\mathsf{I}\left(\frac{n}{2}\right) + 6\mathsf{MM}\left(\frac{n}{2}\right) + Cn^2,$$

C étant une constante. Il suffit alors d'appliquer le lemme « diviser pour régner » avec les choix $m = p = s = 2$, $\kappa = 1$ et $q = 4$. ■

Une première conséquence immédiate du Théorème 8.5 est que la résolution du système linéaire $Ax = b$, pour $b \in \mathbb{K}^n$, et A inversible vérifiant les hypothèses du Théorème 8.5, peut s'effectuer également en $O(\mathsf{MM}(n)) = O(n^\theta)$ opérations dans $\mathbb{K}$.

Déterminant

La factorisation (8.2) montre qu'une légère adaptation de l'algorithme d'inversion permet de calculer le déterminant de A en même temps que son inverse, en la même complexité. Pour cela, il suffit de remplacer les étapes (1), (3), (5) et (7) de l'Algorithme 8.2 par les étapes (1′), (3′), (5′) et (7′) décrites ci-dessous.

1′. Si $n = 1$, renvoyer A^{-1} et A.
3′. Calculer $e := a^{-1}$ et $d_a := \det(a)$ récursivement.
5′. Calculer $t := Z^{-1}$ et $d_Z := \det(Z)$ récursivement.
7′. Renvoyer $\begin{pmatrix} x & y \\ z & t \end{pmatrix}$ et $d_a d_Z$.

Affaiblissement des hypothèses

À ce stade, nous avons démontré les parties (a)–(c) du Théorème 8.4 sous une hypothèse supplémentaire. Dans le cas où $\mathbb{K}$ est un sous-corps du corps des réels $\mathbb{R}$, on peut s'affranchir de cette hypothèse, en observant que, quelle que soit la matrice A inversible dans $\mathcal{M}_n(\mathbb{R})$, la matrice $B = {}^tA \cdot A$ est définie positive et vérifie les hypothèses du Théorème 8.5, et donc l'inverse de A peut se calculer grâce à l'égalité $A^{-1} = B^{-1} \cdot {}^tA$ en $O(\mathsf{MM}(n)) = O(n^\theta)$ opérations dans $\mathbb{K}$.

Il n'est pas difficile de prouver que si la matrice A est de type L (triangulaire inférieure) ou U (triangulaire supérieure) alors l'algorithme de Strassen calcule son inverse sans avoir besoin d'*aucune* hypothèse supplémentaire. Cette remarque est à la base d'un algorithme général d'inversion dû à Bunch et Hopcroft. Cet algorithme calcule d'abord une décomposition LUP d'une matrice inversible arbitraire [4] *sur un corps quelconque* $\mathbb{K}$ en $O(\mathsf{MM}(n))$ opérations, et arrive à en déduire le calcul d'inverse (et aussi la résolution de système) pour le même prix.

L'algorithme d'inversion de Strassen est en outre à la base de toute une algorithmique pour les matrices structurées, qui sera présentée au Chapitre 10.

Résolution des sytèmes surdéterminés

Lorsqu'une matrice n'est pas carrée, le calcul de son noyau se ramène aisément au cas carré, comme illustré dans le résultat suivant qui nous sera utile dans le Chapitre 22 sur la factorisation des polynômes à plusieurs variables.

4. La détection de l'inversibilité est également réalisable en $O(\mathsf{MM}(n))$ opérations, grâce à un algorithme qui calcule le rang de A en cette complexité ; cet algorithme dépasse notre cadre.

Théorème 8.6 Soit $\mathbb{K}$ un corps et supposons qu'on dispose d'un algorithme de multiplication dans $\mathcal{M}_n(\mathbb{K})$ de complexité $\mathsf{MM}(n) = O(n^\theta)$, avec $\theta \geq 2$. Le calcul d'une base du noyau d'une matrice A à n colonnes et $m \geq n$ lignes sur $\mathbb{K}$ coûte $\tilde{O}(mn^{\theta-1})$ opérations dans $\mathbb{K}$.

Démonstration. Le Théorème 8.4 implique ce lemme si $m = n$. Donc, quitte à ajouter des lignes nulles à la matrice A, on peut supposer que m est un multiple de n. Notons A_1 la matrice composée des m premières lignes de A, A_2 celle des m suivantes, etc. On peut calculer une base du noyau de A_1 avec un coût $\tilde{O}(n^\theta)$: notons N_1 la matrice dont les colonnes sont les vecteurs de cette base, de sorte que $A_1N_1 = 0$. On calcule ensuite la matrice N_2 dont les colonnes forment une base du noyau de A_2N_1, de sorte que $A_2N_1N_2 = 0$. On continue de la même manière avec A_3, en calculant N_3 telle que $A_3N_1N_2N_3 = 0$, etc. Finalement les colonnes de $N_1 \cdots N_{m/n}$ forment une base du noyau de A. On obtient ainsi le coût total annoncé $\tilde{O}(mn^{\theta-1})$. ■

Calcul du polynôme caractéristique

Soit A une matrice de $\mathcal{M}_n(\mathbb{K})$. Dans cette section, nous présentons un algorithme efficace, dû à Keller-Gehrig, pour le calcul du polynôme caractéristique $\chi_A(X) = \det(XI_n - A)$ de A. Nous ne décrirons en détail qu'un cas particulier très simple (et très fréquent). Plus exactement, nous ferons dans la suite l'hypothèse que $\chi_A(X) = X^n + p_{n-1}X^{n-1} + \cdots + p_0$ est irréductible dans $\mathbb{K}[X]$; en particulier, il coïncide avec le polynôme minimal de A. L'algorithme repose de façon cruciale sur la propriété suivante.

Lemme 8.7 Soit A une matrice de $\mathcal{M}_n(\mathbb{K})$, dont le polynôme caractéristique est irréductible dans $\mathbb{K}[X]$. Soit v un vecteur de $\mathbb{K}^n \setminus \{0\}$ et soit $P \in \mathcal{M}_n(\mathbb{K})$ la matrice dont la j-ième colonne vaut $A^{j-1}v$, pour $1 \leq j \leq n$. Alors P est inversible et la matrice $P^{-1}AP$ est de type compagnon.

Rappelons qu'une matrice $C = (c_{i,j})_{i,j=1}^n$ de $\mathcal{M}_n(\mathbb{K})$ est appelée *de type compagnon* si ses $n-1$ premières colonnes ne contiennent que des éléments nuls, à l'exception des éléments $c_{2,1}, c_{3,2}, \ldots, c_{n,n-1}$ qui valent tous 1. L'intérêt du Lemme 8.7 vient du fait classique que le polynôme caractéristique de la matrice C vaut $X^n - \sum_{i=1}^n c_{i,n}X^{i-1}$ et n'a donc besoin d'aucune opération arithmétique pour être calculé.

Démonstration. Pour prouver ce lemme, on commence par observer que, grâce à l'hypothèse sur A, la famille $\mathcal{B} = \{v_1, v_2, \ldots, v_n\}$, où $v_i = A^{i-1}v$, forme une base du $\mathbb{K}$-espace vectoriel $\mathbb{K}^n$, et donc P est inversible. En effet, en supposant le contraire, on déduit l'existence d'un polynôme non nul $Q \in \mathbb{K}[X]$ de degré strictement inférieur à n, tel que $Q(A)v = 0$. Sans perte de généralité, ce polynôme peut être supposé de degré minimal avec cette propriété. Comme par ailleurs $\chi_A(A)v = 0$, il s'ensuit que Q divise χ_A ; or, ce dernier étant supposé irréductible, cela entraîne que Q est constant, et donc $v = 0$, une contradiction.

Puisque $Av_{i-1} = v_i$, la matrice C de l'application linéaire $w \mapsto Aw$ dans la base $\mathcal{B}$ est de type compagnon. Le théorème du changement de base fournit alors l'égalité $C = P^{-1}AP$ et permet donc de conclure. ■

Entrée Une matrice A de $\mathcal{M}_n(\mathbb{K})$, avec $n = 2^k$.
Sortie Son polynôme caractéristique $\det(XI_n - A)$.

1. Choisir v dans $\mathbb{K}^n \setminus \{0\}$.
2. $M := A$; $P := v$.
3. Pour i de 1 à k, remplacer P par la concaténation horizontale de P et MP, puis M par M^2 ;
4. $C := P^{-1}AP$.
5. Renvoyer $\left(X^n + p_{n-1}X^{n-1} + \cdots + p_0\right)$, où ${}^t(-p_0, \ldots, -p_{n-1})$ est la dernière colonne de C.

Algorithme 8.3 – Calcul du polynôme caractéristique par l'algorithme de Keller-Gehrig.

Les matrices A et $C = P^{-1}AP$ étant semblables, elles ont le même polynôme caractéristique. L'idée de l'algorithme de Keller-Gehrig est alors naturelle : construire la matrice P du Lemme 8.7, pour ensuite déterminer la matrice C à l'aide d'une multiplication et d'une inversion, et lire les coefficients du polynôme caractéristique de A sur les éléments de la dernière colonne de C.

Du point de vue de la complexité, la seule étape coûteuse est la construction de la matrice P. En effet, la méthode directe qui consiste à calculer la suite (dite *de Krylov*) $v, Av, \ldots, A^{n-1}v$ par multiplications successives d'un vecteur par la matrice A a un coût cubique en n. La remarque cruciale est que l'on peut regrouper les produits matrice-vecteur en plusieurs produits matrice-matrice, de la façon suivante : on calcule en $O(n^2)$ opérations les vecteurs v et Av, on détermine ensuite par *exponentiation binaire* les $O(\log n)$ matrices $A, A^2, A^4, A^8, \ldots$ et on termine le calcul des colonnes de P par le calcul des produits $A^2 \times (v \mid Av) = (A^2v \mid A^3v)$, puis $A^4 \times (v \mid \cdots \mid A^3v) = (A^4v \mid \cdots \mid A^7v)$, etc. Chaque produit de ce type est effectué à l'aide d'un produit matriciel en taille $n \times n$, en rajoutant artificiellement des colonnes nulles aux facteurs droits. Cette méthode conduit à l'Algorithme 8.3. Nous venons de prouver la version suivante plus faible du point (d) du Théorème 8.4 :

Théorème 8.8 Soit $\mathbb{K}$ un corps et supposons qu'on dispose d'un algorithme de multiplication dans $\mathcal{M}_n(\mathbb{K})$ de complexité $\mathsf{MM}(n) = O(n^\theta)$, avec $\theta \geq 2$. Si le polynôme caractéristique d'une matrice A de $\mathcal{M}_n(\mathbb{K})$ est irréductible dans $\mathbb{K}[X]$, l'Algorithme 8.3 le calcule en $O(\mathsf{MM}(n)\log n)$ opérations dans $\mathbb{K}$.

Le calcul de la suite de Krylov sera utilisé au Chapitre 9 dans l'algorithme de Wiedemann pour la résolution des systèmes creux, et de façon moins transparente, au Chapitre 11 dans l'algorithme de Storjohann pour la résolution de systèmes linéaires à coefficients polynomiaux.

Algorithmes sans division

Il est parfois utile de calculer le polynôme caractéristique sans effectuer de division dans l'anneau des coefficients de la matrice. Nous utiliserons en Partie V le résultat suivant, que nous admettrons ici : en comparaison avec le Théorème 8.4, l'évitement

des divisions se fait en perdant 1 sur l'exposant de complexité.

Théorème 8.9 — Berkowitz. Le polynôme caractéristique d'une matrice carrée de taille n à coefficients dans un anneau commutatif $\mathbb{A}$ peut se calculer en $O(n^{\theta+1})$ opérations dans $\mathbb{A}$, où θ est un exposant réalisable pour le produit.

Il est cependant aisé d'obtenir cette même borne de complexité en supposant les divisions par $2,\dots,n$ possibles dans l'anneau des coefficients (ce qui est le cas en présence d'une $\mathbb{Q}$-algèbre par exemple). En effet, la i-ième somme de Newton (définie en Section 3.3) du polynôme caractéristique de la matrice A s'obtient comme $\mathrm{Tr}(A^i)$. Il suffit donc d'utiliser l'Algorithme 3.5 pour obtenir les coefficients du polynôme caractéristique de A à partir de ces traces pour $i = 0,\dots,n-1$. Il s'agit de la *méthode de Le Verrier*.

Exercices

Exercice 8.8 — Multiplication de matrices rectangulaires. Si A et B sont deux matrices $n \times r$ et $r \times k$ avec $n \geq r$ et $k \geq r$, à coefficients dans un corps $\mathbb{K}$, expliquer comment calculer le produit AB en $O(nkr^{\theta-2})$ opérations arithmétiques dans $\mathbb{K}$. ■

Exercice 8.9 — Multiplication et division rapides de nombres complexes.

1. Montrer qu'il est possible d'effectuer le produit matrice-vecteur
$$\begin{pmatrix} a & -b \\ b & a \end{pmatrix} \times \begin{pmatrix} x \\ y \end{pmatrix}$$
en utilisant seulement 3 multiplications scalaires.
2. En déduire que deux nombres complexes peuvent être multipliés en utilisant 3 multiplications réelles.
3. Montrer qu'on peut calculer le quotient de deux nombres complexes en utilisant au plus 7 multiplications et divisions réelles.
4. Donner un algorithme qui utilise seulement 6 opérations de ce type.

■

Exercice 8.10 — Multiplication rapide de quaternions.

1. Soit $A = (a_{ij})$ une matrice $n \times n$ à coefficients dans $\mathbb{K}$, telle que $a_{ij} = \pm a_{ji}$ pour $i < j$. Soit v un vecteur quelconque de $\mathbb{K}^n$. Si ℓ est le nombre d'éléments non nuls de l'ensemble $\{a_{ij}, i < j\}$, montrer que $n + \ell$ multiplications suffisent pour calculer le produit Av.
2. Montrer qu'on peut calculer le produit d'une matrice $n \times n$ et d'une matrice $n \times 2$ en utilisant $\frac{3}{2}n^2 + \frac{5}{2}n$ multiplications scalaires. Indication : Remplacer le produit $M \times (\ v_1 \mid v_2\ )$ par $\begin{pmatrix} 0 & M \\ M & 0 \end{pmatrix} \times \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$, puis décomposer la matrice $A = \begin{pmatrix} 0 & M \\ M & 0 \end{pmatrix}$ en somme de deux matrices vérifiant les hypothèses de la question précédente.
3. Un *quaternion* est une combinaison formelle $a \cdot 1 + b \cdot i + c \cdot j + d \cdot k$, avec $a, b, c, d \in \mathbb{R}$

et i, j, k éléments vérifiant la table de multiplication interne

$\cdot$	1	i	j	k
1	1	i	j	k
i	i	-1	k	$-j$
j	j	$-k$	-1	i
k	k	j	$-i$	-1

(par exemple, $i \cdot j = k$ et $k \cdot j = -i$). La multiplication $\cdot$ se prolonge par $\mathbb{R}$-linéarité à l'ensemble $\mathbb{H}$ des quaternions et en fait un corps non commutatif.
Montrer que deux quaternions arbitraires peuvent être multipliés en utilisant seulement 10 multiplications réelles.

■

Exercice 8.11 Soit P un polynôme de degré au plus n à coefficients dans un corps $\mathbb{K}$. Soit $\theta > 2$ un réel tel que $O(n^\theta)$ opérations dans $\mathbb{K}$ suffisent pour multiplier deux matrices arbitraires de $\mathcal{M}_n(\mathbb{K})$.

1. Donner un algorithme pour évaluer P sur $\lceil \sqrt{n} \rceil$ éléments de $\mathbb{K}$ en $O(n^{\theta/2})$ opérations.
2. Si $A \in \mathcal{M}_n(\mathbb{K})$, montrer qu'on peut calculer la matrice $P(A)$ en $O(n^{\theta+1/2})$ opérations de $\mathbb{K}$.
3. Si $Q \in \mathbb{K}[X]$ est un autre polynôme de degré au plus n, montrer qu'il est possible de calculer les n premiers coefficients du polynôme $P(Q(X))$ en $O(n^{\frac{\theta+1}{2}})$ opérations dans $\mathbb{K}$.

Indication : Écrire P *sous la forme* $P_0(X) + P_1(X)X^d + P_2(X)(X^d)^2 + \cdots$, *avec* d *bien choisi et* $P_i(X)$ *de degrés au plus* $d - 1$. ■

Exercice 8.12 — Inversion de Strassen des matrices polynomiales. Soit $M(X)$ une matrice de $\mathcal{M}_n(\mathbb{K}[X])$ une matrice polynomiale carrée de taille n et de degré borné par d. Exhiber une borne supérieure pour la complexité (exprimée en termes d'opérations arithmétiques dans $\mathbb{K}$ et en fonction des deux paramètres n et d) de l'algorithme de Strassen pour le calcul de $M(X)^{-1}$, sous l'hypothèse que toutes les matrices rencontrées au cours de l'algorithme sont inversibles. ■

Exercice 8.13 — Dérivation automatique et calcul d'inverse. Le but de cet exercice est de démontrer que le calcul de l'inverse d'une matrice n'est pas plus difficile que le calcul du déterminant, en utilisant des techniques de dérivation automatique.

Pour cela, formalisons brièvement ce qu'on entend par *programme*. Dans notre contexte, un programme prend en entrée des indéterminées $Y_1, \ldots, Y_m$ sur un corps $\mathbb{K}$, et effectue une suite d'*instructions* $g_1, \ldots, g_L$. Pour tout $i = 1, \ldots, L$, l'instruction g_i consiste simplement en la définition d'un polynôme G_i, de la forme

$$g_i : \; G_i = \mathsf{Arg}_i \, \mathrm{op}_i \, \mathsf{Arg}'_i,$$

selon les règles suivantes :

— Arg_i et Arg'_i sont dans $\mathbb{K} \cup \{G_1, \ldots, G_{i-1}\} \cup \{Y_1, \ldots, Y_m\}$.

— op_i est dans $\{+,-,\times\}$.

La *taille* d'un tel programme est L. On dit qu'un tel programme *calcule* un polynôme F si F appartient à l'ensemble $\{G_1,\ldots,G_L\}$; on dit de même que le programme calcule $\{F_1,\ldots,F_s\}$ si tous les F_i sont dans $\{G_1,\ldots,G_L\}$.

Par exemple, le programme ayant pour entrées Y_1, Y_2, Y_3, Y_4, pour instructions

$$\begin{array}{rcl} G_1 & = & Y_1 \times Y_4 \\ G_2 & = & Y_2 \times Y_3 \\ G_3 & = & G_1 - G_2 \end{array}$$

calcule le déterminant $Y_1Y_4 - Y_2Y_3$ de la matrice

$$\begin{pmatrix} Y_1 & Y_2 \\ Y_3 & Y_4 \end{pmatrix};$$

il a taille 3.

Si G est dans $\mathbb{K}[Y_1,\ldots,Y_m]$, on appelle *gradient* de G l'ensemble de ses m dérivées partielles $\partial G/\partial Y_i$, $i = 1,\ldots,m$.

1. Soient G et H dans $\mathbb{K}[Y_1,\ldots,Y_m]$. Exprimer les dérivées partielles de $G+H$, $G-H$ et $G\times H$, en fonction de G, H et des dérivées partielles de G et H.
   En déduire qu'étant donné un programme de taille L, qui calcule un polynôme G de $\mathbb{K}[Y_1,\ldots,Y_m]$, on peut en déduire un programme de taille $O(Lm)$ qui calcule G et son gradient.
2. Soient G dans $\mathbb{K}[Y_1,\ldots,Y_m]$ et H dans $\mathbb{K}[Y_1,\ldots,Y_{m+1}]$, tels que
   $$G = H(Y_1,\ldots,Y_m, Y_i \times Y_j),$$
   avec $1 \le i,j \le m$. Exprimer les dérivées partielles de G en fonction de H et de ses dérivées partielles. Traiter ensuite les cas
   $$G = H(Y_1,\ldots,Y_m, Y_i \pm Y_j).$$
3. Soient G dans $\mathbb{K}[Y_1,\ldots,Y_m]$ et H dans $\mathbb{K}[Y_1,\ldots,Y_{m+1}]$, tels que
   $$G = H(Y_1,\ldots,Y_m, Y_i \text{ op } Y_j),$$
   avec $1 \le i,j \le m$, et op dans $\{+,-,\times\}$. On suppose connu un programme de taille L qui calcule H et son gradient. Montrer qu'on peut en déduire un programme de taille $L + O(1)$ qui calcule G et son gradient.
   En déduire par récurrence que si un polynôme G de $\mathbb{K}[Y_1,\ldots,Y_m]$ peut être calculé par un programme de taille L, G et son gradient peuvent être calculés par un programme de taille $O(L)$, où la constante dans le $O(\cdot)$ ne dépend pas du nombre de variables m.

4. Soient $X_{1,1},\dots,X_{1,n},\dots,X_{n,1},\dots,X_{n,n}$ des indéterminées, et soit D le déterminant de la matrice
$$M = \begin{pmatrix} X_{1,1} & \dots & X_{1,n} \\ \vdots & & \vdots \\ X_{n,1} & \dots & X_{n,n} \end{pmatrix}.$$
Montrer que le gradient de D permet de retrouver les entrées de la *comatrice* N de M, où N est l'unique matrice satisfaisant
$$M \cdot {}^tN = {}^tN \cdot M = DI_n,$$
I_n étant la matrice identité de taille n, et ${}^tN$ la transposée de N.
5. Montrer qu'étant donné un programme de taille L qui calcule D, on peut en déduire un programme de taille O(L), effectuant éventuellement des divisions, qui calcule les coefficients de l'inverse de M.
6. En déduire que $\omega_{\text{inv}} \le \omega_{\text{det}}$.

■

Notes

L'optimalité de l'algorithme naïf pour le produit matrice-vecteur mentionnée au début de la Section 8.2 a été prouvée par Winograd [Win67]. Le résultat de l'Exercice 8.9 est dû à Smith [Smi62] ; Alt et van Leeuwen [AL81] et Lickteig [Lic87] ont prouvé que la borne 6 est optimale. Le résultat de l'Exercice 8.10 est dû à Fiduccia [Fid72c] ; la borne 10 peut être améliorée à 8, et celle-ci est optimale [BCS97, (17.32)].

Les parties (b) et (c) du Théorème 8.2 sont dues à Winograd [Win68] et Waksman [Wak70]. Paterson [Pat75] a prouvé que, si l'on s'interdit les soustractions, la multiplication matricielle cubique est optimale. Le Théorème 8.4, dont le titre est emprunté à l'article de Strassen de 1969 [Str69], montre que du point de vue de la complexité on peut faire mieux que la méthode du pivot de Gauss. C'est un résultat surprenant car Klyuyev et Kokovkin–Shcherbak [KK65] avaient prouvé que l'élimination gaussienne est optimale si l'on se restreint à des opérations sur des lignes et des colonnes. C'est surtout un résultat qui a ouvert la voie à tout un domaine de recherche toujours très actif, *la théorie de la complexité bilinéaire*. Les livres de Pan [Pan84b], de Bürgisser, Clausen et Shokrollahi [BCS97, §15–16] et de Abdeljaoued et Lombardi [AL04, §7–10], ainsi que les articles de synthèse de Pan [Pan84a] et de Strassen [Str90] constituent de bonnes références sur ce sujet.

L'interprétation des formules de Strassen donnée en Section 8.2 et l'Exercice 8.6 sont dus à Fiduccia [Fid72c]. L'algorithme d'inversion des matrices génériques en Section 8.3 est dû à Strassen [Str69]. La remarque de Schönhage permettant de traiter l'inversion rapide d'une matrice réelle arbitraire est tirée de [Sch73]. L'algorithme général d'inversion évoqué à la fin de la Section 8.3, ainsi que la preuve des égalités $\omega_{\text{inv}} = \omega_{\text{LUP}} = \omega_{\text{mul}}$ ont été donnés par Bunch et Hopcroft [BH74]. La preuve de l'inégalité $\omega_{\text{inv}} \le \omega_{\text{det}}$ esquissée dans l'Exercice 8.13 est due à Baur et Strassen [BS83]. La réduction présentée en Section 8.3 provient de l'article de Munro [Mun73]. La partie (f) du Théorème 8.4 est due à Keller-Gehrig [Kel85], et la partie (g) est due à

Bürgisser, Karpinski et Lickteig [BKL91]. Le fait que l'exposant ω_{mul} ne dépend que de la caractéristique du corps de base a été prouvé par Schönhage [Sch81].

L'algorithme décrit dans la Section 8.3 a été conçu par Keller-Gehrig [Kel85] ; il s'agit d'une version accélérée d'un algorithme classique de Krylov et Danilevskii [Dan37 ; Kry31]. Il est possible d'adapter cet algorithme au cas où A est une matrice arbitraire. Plus précisément, l'algorithme de Keller-Gehrig calcule dans ce cas une base de $\mathbb{K}^n$ dans laquelle la matrice de l'endomorphisme de multiplication par A est triangulaire par blocs, avec des blocs diagonaux de type compagnon. Autrement dit, il procède à une mise en *forme de Frobenius faible*. Un raffinement de cet algorithme, dû à Giesbrecht [Gie95], permet de calculer la *forme de Frobenius forte* (matrice diagonale par blocs de type compagnon) également en complexité $\tilde{\mathrm{O}}(\mathsf{MM}(n))$; cet algorithme permet aussi le calcul, en la même complexité, du polynôme minimal et de tous les autres facteurs invariants de A. Les algorithmes de Giesbrecht [Gie95] sont probabilistes ; des versions déterministes ont été données par Storjohann [Sto01]. Keller-Gehrig [Kel85] propose également un algorithme différent de celui décrit dans la Section 8.3, calculant le polynôme caractéristique d'une matrice *générique* en seulement $\mathrm{O}(\mathsf{MM}(n))$ opérations ; voir l'Exercice 33.11 en page 599. Une version (probabiliste) de même complexité et sans hypothèse de généricité a été proposée par Pernet et Storjohann [PS07].

Giesbrecht [Gie95] et Storjohann [Sto01] montrent que $\omega_{\text{Frobenius}} = \omega_{\text{mul}}$ et que l'évaluation d'un polynôme de degré d en une matrice de $\mathcal{M}_n(\mathbb{K})$ peut se faire en $\tilde{\mathrm{O}}(\mathsf{MM}(n) + d)$ opérations dans $\mathbb{K}$, ce qui améliore le résultat (2) de l'Exercice 8.11. Cet exercice est inspiré des articles de Borodin et Munro [BM71] pour (1), de Paterson et Stockmeyer [PS73] pour (2) et de Brent et Kung [BK78] pour (3) ; le résultat de (1) sera amélioré au Chapitre 5 et celui de (3) au Chapitre 3.

À l'heure actuelle, on ne sait pas si les exposants des problèmes suivants

— résoudre un système ;
— calculer le rang ;
— décider si une matrice est inversible ;

sont eux aussi égaux à ω_{mul}. Grâce au Théorème 8.4, on sait que ces problèmes ne sont pas plus difficiles que la multiplication, mais peut-être sont-ils plus faciles …

Un autre problème ouvert est le calcul du polynôme caractéristique par un algorithme déterministe de complexité $\mathrm{O}(\mathsf{MM}(n))$.

L'algorithme de Berkowitz a été décrit pour la première fois en 1984 [Ber84]. L'algorithme de Le Verrier se trouve dans son article de 1840 [Le 40].

Complexité des algorithmes commutatifs

Il est naturel de se demander s'il est possible d'obtenir des algorithmes *à la Winograd* plus efficaces, en regroupant les termes de (8.1) en paquets plus gros plutôt que deux par deux. La réponse est négative. Harter [Har72] a montré qu'il n'existe pas de généralisation de (8.1) de la forme

$$(\ell|c) = \lambda(a_1, x_1, a_2, x_2, \ldots) + \mu(a_1, x_1, a_2, x_2, \ldots) + f(a_1, a_2, \ldots) + g(x_1, x_2, \ldots),$$

où λ et μ sont des formes linéaires et f et g des fonctions quelconques. Ja'Ja [JaJ79] a montré plus généralement qu'aucun algorithme commutatif ne peut apporter un gain d'un facteur supérieur à 2 sur l'algorithme naïf, et en particulier ne peut pas

améliorer l'exposant 3. De ce point de vue, les algorithmes de Winograd et Waksman sont optimaux dans la classe des algorithmes commutatifs.

Algorithme de Strassen

Winograd [FP74] a donné une version de l'algorithme de Strassen utilisant seulement C = 15 additions et soustractions (au lieu de C = 18). Cela permet de descendre la constante dans le $O(n^{\log_2 7})$ de 7 à 6 pour l'Algorithme 8.1. Cette constante a été abaissée à 3,92 par Fischer [Fis74].

Dans le cas où la taille n de la matrice n'est pas une puissance de 2, il vaut mieux éviter d'appliquer l'algorithme de Strassen à la matrice augmentée en taille la puissance de 2 supérieure à n, comme le montre l'Exercice 8.4. D'autres stratégies sont décrites par Huss–Lederman, Jacobson, Tsao, Turnbull et Johnson [Hus+96].

Bornes inférieures et bornes supérieures

Le problème des bornes inférieures est très difficile. En taille $n = 2$, il a été montré [HK71 ; Win71] que le résultat de Strassen est optimal du point de vue des multiplications : on ne peut pas effectuer le produit de deux matrices carrées de taille 2 en seulement 6 multiplications scalaires. Récemment Landsberg [Lan06] a obtenu une généralisation de ce résultat, en utilisant des outils géométriques. Probert [Pro76] a montré que la version de l'algorithme de Strassen due à Winograd est optimale également vis-à-vis du nombre total d'opérations arithmétiques : il n'existe aucun algorithme pour multiplier deux matrices 2×2 en 7 multiplications et seulement 14 additions et soustractions.

La taille $n = 2$ est la seule pour laquelle on connaît le nombre minimal de multiplications scalaires requises pour effectuer le produit de matrices carrées $n \times n$. En taille $n = 3$, la meilleure borne inférieure connue à ce jour est 19 [Blä03], alors que le meilleur algorithme commutatif utilise 22 multiplications [Mak86] et le meilleur algorithme non commutatif utilise 23 multiplications [Lad76]. Pour une taille arbitraire, toutes les bornes inférieures connues sont seulement quadratiques en n : les meilleures, $\frac{5}{2}n^2 - 3n$ et $3n^2 - o(n^2)$, sont dues à Bläser [Blä01] et Landsberg [Lan14]. Si le corps des scalaires est $\mathbb{R}$ ou $\mathbb{C}$, et dans un modèle restreint de complexité où les seules multiplications scalaires permises se font par des scalaires de module borné, Raz [Raz03] a prouvé une borne inférieure en $c\,n^2 \log n$.

Matrices de petite taille

Il existe un grand nombre d'autres schémas *à la Strassen* pour multiplier des matrices de petite taille. Un recensement des meilleurs algorithmes non commutatifs connus pour le produit des matrices de petite taille a été effectué par Smith [Smi02].

Par exemple, Sýkora [Sýk77] a montré qu'on peut multiplier deux matrices $n \times n$ en $n^3 - (n-1)^2$ produits non commutatifs ; cet algorithme généralise à la fois ceux de Strassen et de Laderman. Des techniques plus sophistiquées ont permis à Pan [Pan78 ; Pan80] de montrer que le produit de matrices $n \times n$ peut s'effectuer en $n^3/3 + 9n^2/2 - n/3$ multiplications non commutatives. Chacun de ces schémas, appliqué récursivement, donne un algorithme de produit matriciel sous-cubique. Par exemple, Pan [Pan80] montre que l'on peut multiplier des matrices 48×48 en 47216 multiplications non commutatives, ce qui mène à $\omega_{\text{mul}} < 2{,}781$.

À la recherche de $\omega_{\mathbf{mul}} = 2$

L'idée initiale de Strassen de combiner une bonne multiplication en taille fixée et la puissance de la récursivité pour améliorer l'exposant ω_{mul} n'a pas permis d'aller en dessous de 2,7. Les améliorations ultérieures reposent toutes sur une accumulation d'idées de plus en plus sophistiquées. Une première idée fut d'utiliser le produit des matrices rectangulaires. Si l'on dispose d'un algorithme pour multiplier deux matrices $(m\times n)\times(n\times p)$ en ℓ multiplications, alors on sait effectuer en ℓ multiplications chacun des produits en tailles $(n\times p)\times(p\times m)$ et $(p\times m)\times(m\times n)$ [HM73 ; Pan72]. Cet résultat est lié au théorème de transposition de Tellegen qui sera vu au Chapitre 12. Une décomposition par blocs permet alors d'obtenir un algorithme pour le produit de matrices carrées de taille mnp en ℓ^3 opérations, et d'obtenir ainsi la majoration $\omega_{\text{mul}} \leq 3\log(\ell)/\log(mnp)$. Ce résultat n'a permis d'améliorer la majoration de ω_{mul} que combiné avec d'autres concepts, comme les *algorithmes approchés* et le *rang tensoriel limite* (*border rank* en anglais) introduits par Bini et ses collaborateurs [Bin+79 ; Bin80]. L'article de Bini, Capovani, Romani et Lotti [Bin+79] donne un algorithme approché qui utilise 10 multiplications pour le produit matriciel en taille $(2\times 2)\times(2\times 3)$; cela implique $\omega_{\text{mul}} < \log_{12}(1000) < 2{,}77989$, mais les constantes dans les $O(\cdot)$ commencent à devenir astronomiques. D'autres idées pour obtenir des bornes de plus en plus fines sur ω_{mul} sont : l'utilisation des matrices creuses [Sch81] qui aboutit à l'inégalité asymptotique pour les sommes directes de *matrices à trous* et à la majoration $\omega_{\text{mul}} < 2{,}548$; la théorie de la déformation des modules sur des groupes algébriques linéaires conduisant à la *méthode laser* de Strassen [Str87] et à la borne $\omega_{\text{mul}} < 2{,}4785$, et qui est à la base de la borne $\omega_{\text{mul}} < 2{,}3729269$ due à Vassilevska Williams [Vas12 ; Vas14], et améliorant le précédent record dû à Coppersmith et Winograd [CW90]. La meilleure borne actuellement connue, $\omega_{\text{mul}} < 2{,}3728639$, est due à Le Gall [Le 14].

Théorie et pratique

Théoriquement, le principal problème ouvert est $\omega_{\text{mul}} = 2$. D'un point de vue pratique, il n'est pas forcément intéressant de comparer les algorithmes en fonction de leur exposant, mais plutôt de répondre à des questions du type : quelle est la formule la plus simple qui mène à un exposant réalisable $\theta < \log_2 7$? Quel est l'algorithme le plus rapide pour une taille donnée ?

L'idée des algorithmes mentionnés en Section 8.2 est due à Pan [Pan72 ; Pan81]. L'article de Laderman, Pan et Sha [LPS92] est le premier à avoir mis en avant leur aspect pratique ; les derniers progrès basés sur cette idée sont dûs à Kaporin [Kap04 ; Kap99].

La recherche théorique pour améliorer les bornes sur ω_{mul} a demandé beaucoup d'ingéniosité (et de chance !), mais a mené à des algorithmes de plus en plus éloignés du calcul réel. Smith [Smi02] adopte un point de vue pragmatique et propose d'automatiser la recherche d'algorithmes rapides en petite taille. Il ramène la recherche d'un algorithme qui calcule le produit de deux matrices de types $(a\times b)$ et $(b\times c)$ en m multiplications non commutatives à un problème d'optimisation dans un espace à $(ab+bc+ca)m$ dimensions. Cela lui permet de retrouver tous les algorithmes connus pour les petites tailles et d'en découvrir des nouveaux. Ses programmes retrouvent les formules de Strassen en quelques secondes et celles de Laderman en quelques minutes.

Un point de vue différent est celui développé par Landsberg [Lan08], qui montre que les questions ouvertes concernant la complexité de la multiplication matricielle se formulent convenablement en termes géométriques et théorie de la représentation.

Enfin, une autre piste actuellement explorée pour prouver (ou infirmer) que $\omega_{\text{mul}} = 2$ est celle ouverte par Cohn et Umans [Coh+05 ; CU03]. Ils développent une approche unifiée permettant de ramener la multiplication matricielle à l'étude de la transformée de Fourier discrète dans des algèbres de groupe associées à certains groupes finis, possédant de bonnes propriétés vis-à-vis de leurs représentations irréductibles. Ils retrouvent par ce biais $\omega_{\text{mul}} < 2{,}41$ et ramènent $\omega_{\text{mul}} = 2$ à des conjectures combinatoires et en théorie des groupes.

Déterminant et permanent

Le permanent d'une matrice $A \in \mathcal{M}_n(\mathbb{K})$ est défini comme

$$\operatorname{perm}(A) = \sum_{\sigma \in \mathcal{S}_n} \prod_{i=1}^{n} a_{i,\sigma(i)}.$$

Malgré la ressemblance entre cette définition et celle de det(A), on ne connaît pas d'algorithme de complexité polynomiale en n calculant perm(A) sur un corps $\mathbb{K}$ de caractéristique différente de 2. Valiant [Val79] a prouvé que le calcul du permanent pour les matrices $n \times n$ à coefficients dans $\{0, 1\}$ est un problème NP-difficile.

Bibliographie

AL04 Abdeljaoued, J. et H. Lombardi (2004). *Méthodes matricielles : introduction à la complexité algébrique*. Vol. 42. Mathématiques & Applications. Springer-Verlag.

AL81 Alt, H. et J. van Leeuwen (1981). « The complexity of basic complex operations ». In : *Computing. Archiv für Informatik und Numerik*, vol. 27, n°3, p. 205–215.

BCS97 Bürgisser, Peter, Michael Clausen et M. Amin Shokrollahi (1997). *Algebraic complexity theory*. Vol. 315. Grundlehren der Mathematischen Wissenschaften. Springer-Verlag.

Ber84 Berkowitz, Stuart J. (1984). « On computing the determinant in small parallel time using a small number of processors ». In : *Information Processing Letters*, vol. 18, p. 147–150.

BH74 Bunch, James R. et John E. Hopcroft (1974). « Triangular factorization and inversion by fast matrix multiplication ». In : *Mathematics of Computation*, vol. 28, p. 231–236.

Bin+79 Bini, D., M. Capovani, F. Romani et G. Lotti (1979). « $O(n^{2.7799})$ complexity for $n \times n$ approximate matrix multiplication ». In : *Information Processing Letters*, vol. 8, n°5, p. 234–235.

Bin80 Bini, D. (1980). « Relations between exact and approximate bilinear algorithms. Applications ». In : *Calcolo*, vol. 17, n°1, p. 87–97.

BK78 Brent, R. P. et H. T. Kung (1978). « Fast algorithms for manipulating formal power series ». In : *Journal of the ACM*, vol. 25, n°4, p. 581–595.

BKL91 Bürgisser, P., M. Karpinski et T. Lickteig (1991). « Some computational problems in linear algebra as hard as matrix multiplication ». In : *Computational Complexity*, vol. 1, n°2, p. 131–155.

Blä01 Bläser, Markus (2001). « A $\frac{5}{2}n^2$-lower bound for the multiplicative complexity of $n \times n$-matrix multiplication ». In : *STACS 2001 (Dresden)*. Vol. 2010. Lecture Notes in Computer Science. Springer-Verlag, p. 99–109.

Blä03 — (2003). « On the complexity of the multiplication of matrices of small formats ». In : *Journal of Complexity*, vol. 19, n°1, p. 43–60.

BM71 Borodin, A. et I. Munro (1971). « Evaluating polynomials at many points ». In : *Information Processing Letters*, vol. 1, n°2, p. 66–68.

BS83 Baur, W. et V. Strassen (1983). « The complexity of partial derivatives ». In : *Theoretical Computer Science*, vol. 22, p. 317–330.

Coh+05 Cohn, H., R. Kleinberg, B. Szegedy et C. Umans (2005). « Group-theoretic algorithms for matrix multiplication ». In : *FOCS'05 : IEEE Conference on Foundations of Computer Science*. IEEE Computer Society, p. 379–388.

CU03 Cohn, Henry et Christopher Umans (2003). « A group-theoretic approach to fast matrix multiplication ». In : *FOCS'03 : IEEE Conference on Foundations of Computer Science*. Washington, DC, USA : IEEE Computer Society, p. 438.

CW90 Coppersmith, Don et Shmuel Winograd (1990). « Matrix multiplication via arithmetic progressions ». In : *Journal of Symbolic Computation*, vol. 9, n°3, p. 251–280.

Dan37 Danilevskii, A. M (1937). « The numerical solution of the secular equation ». In : *Matematicheskiĭ Sbornik*, vol. 44, n°2. (in Russian), p. 169–171.

Fid72c Fiduccia, Charles M. (1972). « On obtaining upper bounds on the complexity of matrix multiplication ». In : *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*. New York : Plenum, p. 31–40.

Fis74 Fischer, Patrick C. (1974). « Further schemes for combining matrix algorithms ». In : *Automata, languages and programming*. Vol. 14. Lecture Notes in Computer Science. Springer-Verlag, p. 428–436.

FP74 Fischer, P. C. et R. L. Probert (1974). « Efficient procedures for using matrix algorithms ». In : *Automata, languages and programming*. Vol. 14. Lecture Notes in Computer Science. Springer-Verlag, p. 413–427.

Gie95 Giesbrecht, Mark (1995). « Nearly optimal algorithms for canonical matrix forms ». In : *SIAM Journal on Computing*, vol. 24, n°5, p. 948–969.

Har72 Harter, Richard (1972). « The optimality of Winograd's formula ». In : *Communications of the ACM*, vol. 15, n°5, p. 352.

HK71 Hopcroft, J. E. et L. R. Kerr (1971). « On minimizing the number of multiplications necessary for matrix multiplication ». In : *SIAM Journal on Applied Mathematics*, vol. 20, p. 30–36.

HM73 Hopcroft, J. et J. Musinski (1973). « Duality applied to the complexity of matrix multiplication and other bilinear forms ». In : *SIAM Journal on Computing*, vol. 2, p. 159–173.

Hus+96 Huss-Lederman, S., E. M. Jacobson, A. Tsao, T. Turnbull et J. R. Johnson (1996). « Implementation of Strassen's algorithm for matrix multiplica-

tion ». In : *Supercomputing'96*. 32 pp. Pittsburgh, PA : IEEE Computer Society.

JaJ79 Ja'Ja', Joseph (1979). « On the complexity of bilinear forms with commutativity ». In : *STOC'79 : ACM Symposium on Theory of Computing*. Atlanta, Georgia, United States : ACM, p. 197–208.

Kap04 Kaporin, I. (2004). « The aggregation and cancellation techniques as a practical tool for faster matrix multiplication ». In : *Theoretical Computer Science*, vol. 315, n°2-3, p. 469–510.

Kap99 Kaporin, Igor (1999). « A practical algorithm for faster matrix multiplication ». In : *Numerical Linear Algebra with Applications*, vol. 6, n°8, p. 687–700.

Kel85 Keller-Gehrig, Walter (1985). « Fast algorithms for the characteristic polynomial ». In : *Theoretical Computer Science*, vol. 36, n°2-3, p. 309–317.

KK65 Klyuyev, V. V. et N. I. Kokovkin-Shcherbak (1965). « On the minimization of the number of arithmetic operations for the solution of linear algebraic systems of equations ». In : *USSR Computational Mathematics and Mathematical Physics*, vol. 5, p. 25–43.

Kry31 Krylov, A. N. (1931). « On the numerical solution of the equation by which in technical questions frequencies of small oscillations of material systems are determined ». In : *Izvestiya Akademii Nauk SSSR*, vol. 7, n°4. (in Russian), p. 491–539.

Lad76 Laderman, Julian D. (1976). « A noncommutative algorithm for multiplying 3×3 matrices using 23 muliplications ». In : *Bulletin of the American Mathematical Society*, vol. 82, n°1, p. 126–128.

Lan06 Landsberg, J. M. (2006). « The border rank of the multiplication of 2×2 matrices is seven ». In : *Journal of the American Mathematical Society*, vol. 19, n°2, p. 447–459.

Lan08 — (2008). « Geometry and the complexity of matrix multiplication ». In : *Bulletin of the American Mathematical Society*, vol. 45, n°2, p. 247–284.

Lan14 — (2014). « New lower bounds for the rank of matrix multiplication ». In : *SIAM Journal on Computing*, vol. 43, n°1, p. 144–149.

Le 14 Le Gall, François (2014). « Powers of tensors and fast matrix multiplication ». In : *ISSAC'14 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Katsusuke Nabeshima. New York, NY, USA : ACM, p. 296–303.

Le 40 Le Verrier, Urbain (1840). « Sur les variations séculaires des éléments elliptiques des sept planètes principales : Mercure, Vénus, la Terre, Mars, Jupiter, Saturne et Uranus ». In : *Journal de Mathématiques Pures et Appliquées*, vol. 1 (5), p. 220–254.

Lic87 Lickteig, Thomas (1987). « The computational complexity of division in quadratic extension fields ». In : *SIAM Journal on Computing*, vol. 16, n°2, p. 278–311.

LPS92 Laderman, Julian, Victor Pan et Xuan He Sha (1992). « On practical algorithms for accelerated matrix multiplication ». In : *Linear Algebra and its Applications*, vol. 162/164, p. 557–588.

Mak86 Makarov, O. M. (1986). « An algorithm for multiplying 3×3 matrices ». In : *USSR Computational Mathematics and Mathematical Physics*, vol. 26, n°1, p. 179–180.

Mun73 Munro, I. (1973). « Problems related to matrix multiplication ». In : *Proceedings Courant Institute Symposium on Computational Complexity, October 1971*. Éd. par R. Rustin. New York : Algorithmics Press, p. 137–151.

Pan72 Pan, V. (1972). « Computation schemes for a product of matrices and for the inverse matrix ». In : *Akademiya Nauk SSSR i Moskovskoe Matematicheskoe Obshchestvo. Uspekhi Matematicheskikh Nauk*, vol. 27, n°5(167), p. 249–250.

Pan78 Pan, V. Ya. (1978). « Strassen's algorithm is not optimal. Trilinear technique of aggregating, uniting and canceling for constructing fast algorithms for matrix operations ». In : *SFCS '78*. Washington, DC, USA : IEEE Computer Society, p. 166–176.

Pan80 — (1980). « New fast algorithms for matrix operations ». In : *SIAM Journal on Computing*, vol. 9, n°2, p. 321–342.

Pan81 — (1981). « New combinations of methods for the acceleration of matrix multiplication ». In : *Computers & Mathematics with Applications. An International Journal*, vol. 7, n°1, p. 73–125.

Pan84a Pan, Victor (1984). « How can we speed up matrix multiplication ? » In : *SIAM Review*, vol. 26, n°3, p. 393–415.

Pan84b — (1984). *How to multiply matrices faster*. Vol. 179. Lecture Notes in Computer Science. Springer-Verlag.

Pat75 Paterson, Michael S. (1975). « Complexity of monotone networks for Boolean matrix product ». In : *Theoretical Computer Science*, vol. 1, n°1, p. 13–20.

Pro76 Probert, Robert L. (1976). « On the additive complexity of matrix multiplication ». In : *SIAM Journal on Computing*, vol. 5, n°2, p. 187–203.

PS07 Pernet, Clément et Arne Storjohann (2007). « Faster algorithms for the characteristic polynomial ». In : *ISSAC'07 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Dongming Wang. ACM Press, p. 307–314.

PS73 Paterson, M. S. et L. J. Stockmeyer (1973). « On the number of nonscalar multiplications necessary to evaluate polynomials ». In : *SIAM Journal on Computing*, vol. 2, n°1, p. 60–66.

Raz03 Raz, Ran (2003). « On the complexity of matrix product ». In : *SIAM Journal on Computing*, vol. 32, n°5, p. 1356–1369.

Sch73 Schönhage, A. (1972/73). « Unitäre Transformationen grosser Matrizen ». In : *Numerische Mathematik*, vol. 20, p. 409–417.

Sch81 — (1981). « Partial and total matrix multiplication ». In : *SIAM Journal on Computing*, vol. 10, n°3, p. 434–455.

Smi02 Smith, Warren D. (2002). « Fast matrix multiplication formulae : report of the prospectors ». Preprint. `http://www.math.temple.edu/~wds/prospector.pdf`.

Smi62 Smith, R. L. (1962). « Algorithm 116 : Complex division ». In : *Communications of the ACM*, n°5, p. 435.

Sto01 Storjohann, Arne (2001). « Deterministic computation of the Frobenius form ». In : *FOCS'01 : IEEE Symposium on Foundations of Computer Science*. Extended abstract. Las Vegas, NV : IEEE Computer Society, p. 368–377.

Str69 Strassen, V. (1969). « Gaussian elimination is not optimal ». In : *Numerische Mathematik*, vol. 13, p. 354–356.

Str87 — (1987). « Relative bilinear complexity and matrix multiplication ». In : *Journal für die Reine und Angewandte Mathematik*, vol. 375/376, p. 406–443.

Str90 Strassen, Volker (1990). « Algebraic complexity theory ». In : *Handbook of theoretical computer science, Vol. A*. Amsterdam : Elsevier, p. 633–672.

Sýk77 Sýkora, Ondrej (1977). « A fast non-commutative algorithm for matrix multiplication ». In : *Mathematical foundations of computer science (Proc. Sixth Sympos., Tatranská Lomnica, 1977)*. Vol. 53. Lecture Notes in Computer Science. Springer-Verlag, p. 504–512.

Val79 Valiant, L. G. (1979). « The complexity of computing the permanent ». In : *Theoretical Computer Science*, vol. 8, n°2, p. 189–201.

Vas12 Vassilevska Williams, Virginia (2012). « Multiplying matrices faster than Coppersmith–Winograd ». In : *STOC'12 : ACM Symposium on Theory of Computing*. New York, NY : ACM, p. 887–898.

Vas14 — (2014). « Multiplying matrices in $O(n^{2.373})$ time ». `http://theory.stanford.edu/~virgi/matrixmult-f.pdf`.

Wak70 Waksman, Abraham (1970). « On Winograd's algorithm for inner products ». In : *IEEE Transactions on Computers*, vol. C-19, n°4, p. 360–361.

Win67 Winograd, S. (1967). « On the number of multiplications required to compute certain functions ». In : *Proceedings of the National Academy of Sciences of the United States of America*, vol. 58, p. 1840–1842.

Win68 — (1968). « A new algorithm for inner-product ». In : *IEEE Transactions on Computers*, vol. 17, p. 693–694.

Win71 — (1971). « On multiplication of 2×2 matrices ». In : *Linear Algebra and Applications*, vol. 4, p. 381–388.

9. Algèbre linéaire creuse : algorithme de Wiedemann

Résumé

Les matrices creuses sont les matrices contenant beaucoup d'éléments nuls. L'algorithme de Wiedemann est une méthode itérative pour résoudre des systèmes linéaires représentés par des matrices creuses. Il ramène la résolution au calcul du polynôme minimal, lui-même reposant sur la reconnaissance d'une suite récurrente à coefficients constants.

9.1 Introduction

Dans ce chapitre, on aborde des questions radicalement différentes de celles du Chapitre 8 : on ne cherche plus à effectuer les produits, inversions, ... de matrices quelconques, mais à manipuler des matrices *creuses*.

On ne commencera pas par définir précisément ce qu'est une matrice creuse. L'approche consiste à donner des algorithmes dont la complexité s'exprime en fonction du nombre d'éléments non nuls des matrices en entrée. De la complexité des algorithmes « creux » va découler une borne, typiquement de l'ordre $O(n)$ pour des matrices de taille $n \times n$, sur le nombre de coefficients non nuls pour que le changement de modèle soit pertinent.

Dans tout ce qui suit, on considère donc une matrice A de taille $n \times n$ à coefficients dans un corps $\mathbb{K}$ et vérifiant les hypothèses suivantes :

— A est inversible,

— A contient s éléments non nuls.

Le produit de A par un vecteur peut donc s'effectuer en $O(s)$ opérations dans $\mathbb{K}$.

On va décrire un algorithme dû à Wiedemann qui permet de calculer l'unique solution du système $Ax = y$ avec une complexité en $O(ns)$; l'algorithme original

est plus général que celui présenté ici, en ce qu'il permet de traiter les matrices rectangulaires ou carrées et non inversibles, mais il se ramène au cas carré inversible.

Remarquons que si s est de l'ordre de n^2, caractérisant donc des matrices plutôt denses, on retombe dans une complexité de l'ordre de $O(n^3)$. Le cas intéressant est celui où s est de l'ordre de n, auquel cas l'algorithme est quadratique en n : pour simplifier, on retiendra que l'exposant de l'algèbre linéaire creuse est 2 dans les bons cas.

9.2 Polynôme minimal et résolution de systèmes

L'algorithme de Wiedemann passe par le calcul du *polynôme minimal* de A. Rappelons sa définition. Il est possible d'associer à tout polynôme en une variable $P = \sum_i p_i X^i$ de $\mathbb{K}[X]$ la matrice $P(A) = \sum_i p_i A^i$. Le théorème de Cayley–Hamilton affirme que le polynôme caractéristique χ_A de A annule la matrice A, c'est-à-dire que $\chi_A(A) = 0$.

Définition 9.1 Le polynôme minimal de A est le polynôme unitaire de plus petit degré annulant A.

Soit $\mu_A(X)$ le polynôme minimal de A, supposé connu. Puisqu'on a supposé A inversible, le terme constant de $\mu_A(X)$ n'est pas nul : il divise le terme constant du polynôme caractéristique, qui n'est lui-même pas nul.

L'égalité $\mu_A(A) = 0$ se récrit sous la forme

$$A^{-1} = \frac{-1}{p_0}\left(A^{m-1} + p_{m-1}A^{m-2} + \cdots + p_1 I_n\right).$$

Pour résoudre le système $Ax = b$, $(b \in \mathbb{K}^n)$, on va calculer $x = A^{-1}b$, c'est-à-dire

$$\frac{-1}{p_0}\left(A^{m-1}b + p_{m-1}A^{m-2}b + \cdots + p_1 b\right).$$

Ainsi, il suffit de calculer les itérés $b, Ab, \ldots, A^{m-1}b$, puis d'additionner les vecteurs $p_1 b, p_2 Ab, \ldots, A^{m-1}b$, pour retrouver x. L'analyse de complexité est immédiate :

— Chacun des $A^i b$ s'obtient à partir de $A^{i-1}b$ en $O(s)$ opérations.
— Chacun des produits par p_i, et chaque addition, se fait en $O(n)$ opérations. Remarquer que $n \leq s$ (hypothèse d'inversibilité de A).

Au total, on a donc $O(ns)$ opérations à faire pour résoudre le système, si on connaît le polynôme minimal de A.

9.3 Calcul du polynôme minimal

Écrivons le polynôme minimal de A sous la forme

$$\mu_A(X) = X^m + p_{m-1}X^{m-1} + \cdots + p_0.$$

Alors la suite des puissances de A vérifie la récurrence linéaire

$$A^{k+m} + p_{m-1}A^{k+m-1} + \cdots + p_0 A^k = 0,$$

Entrée Une matrice creuse $A \in \mathcal{M}_n(\mathbb{K})$.
Sortie Son polynôme minimal $\mu_A(X)$.
1. Choisir des vecteurs u et v aléatoires dans $\mathbb{K}^n$.
2. Pour $0 \leq i \leq 2n$, calculer les vecteurs $v_i = A^i v$, puis les scalaires $N_i = {}^t u\, v_i$.
3. Renvoyer le polynôme minimal de la suite (N_i).

Algorithme 9.1 – Algorithme de Wiedemann pour calculer le polynôme minimal d'une matrice creuse.

et ne vérifie aucune récurrence d'ordre plus petit.

Pour tous vecteurs u et v, la suite (de nombres) $N_i := {}^t u A^i v$ vérifie donc

$$N_{k+m} + p_{m-1} N_{k+m-1} + \cdots + p_0 N_k = 0.$$

Si u et v sont mal choisis (nuls, par exemple), la suite N_i vérifie une récurrence d'ordre plus petit que m. La proposition suivante montre qu'en choisissant u et v au hasard, on tombe vraisemblablement sur une suite N_i qui ne satisfait pas de récurrence d'ordre plus petit.

Proposition 9.1 Il existe un polynôme $D \in \mathbb{K}[U_1, \ldots, U_n, V_1, \ldots, V_n]$, non nul et de degré au plus $2n$, tel que si $D(u, v)$ est non nul, la suite N_i associée à u et v ne satisfait pas de récurrence d'ordre plus petit que m.

L'idée est de choisir u et v au hasard. Si on n'a pas de chance, $D(u, v)$ est nul ; sinon, la suite des N_i associée à u et v permet de retrouver $\mu_A(X)$ grâce à l'algorithme de Berlekamp–Massey décrit en page 143 (attention, on ne connaît pas explicitement le polynôme D, mais son existence assure que la plupart des choix de u et v sont « chanceux »).

L'Algorithme 9.1 résume cette approche probabiliste. L'analyse de probabilité est facile à faire en utilisant les résultats de l'Exercice 9.1 ci-dessous.

La complexité de l'étape (2) est de $O(n)$ produits matrice-vecteur ou vecteur-vecteur, chacun prenant au pire $O(s)$ opérations ; au total on obtient donc $O(ns)$ opérations pour cette phase. L'algorithme d'approximants de Padé est négligeable, puisqu'il ne demande que $O(n^2) = O(sn)$ opérations.

9.4 Calcul du déterminant

L'Algorithme 9.1 permet de détecter en cours de calcul l'inversibilité de la matrice A. En effet, si l'approximant de Padé calculé par l'algorithme de Berlekamp–Massey (page 143) a un dénominateur divisible par X, cela veut dire que $\det(A) = 0$.

Il est possible de calculer en bonne complexité le déterminant de la matrice creuse A, en exploitant le résultat suivant d'algèbre linéaire.

Lemme 9.2 Si tous les mineurs principaux de $A \in \mathcal{M}_n(\mathbb{K})$ sont non nuls, alors les polynômes minimal et caractéristique de la matrice $B = A \cdot Diag(X_1, \ldots, X_n)$ coïncident.

L'algorithme qui s'en déduit est le suivant :
- Choisir une matrice de permutations aléatoire P, pour assurer que tous les mineurs principaux de la matrice AP sont inversibles ;
- choisir des éléments $x_1, \ldots, x_n$ aléatoirement dans $\mathbb{K}$;
- calculer le polynôme minimal μ_B de $B = A \cdot P \cdot \mathrm{Diag}(x_1, \ldots, x_n)$;
- renvoyer $\mu_B(0)/(x_1 \cdots x_n) \cdot \mathrm{sgn}(P)$.

À nouveau, la complexité de cet algorithme est quadratique en la taille n de A.

9.5 Calcul du rang

Le rang maximal de la matrice creuse A est détecté par les algorithmes précédents. Quitte à multiplier A par une matrice de permutations aléatoires, on peut donc supposer que son rang r est strictement inférieur à n, et que ses mineurs principaux A_i, pour $1 \leq i \leq r$, sont tous non nuls.

Sous ces hypothèses, il est possible de montrer que, pour un choix aléatoire d'éléments $x_1, \ldots, x_n$ dans $\mathbb{K}$, le rang r est égal à $\deg(\mu_{AD}) - 1$, où D est la matrice diagonale $D = \mathrm{Diag}(x_1, \ldots, x_n)$. L'algorithme qui s'ensuit calcule le polynôme minimal de la matrice creuse AD, et en déduit le rang de A, le tout pour une complexité quadratique en n.

Exercices

Exercice 9.1 Soit A une matrice dans $\mathcal{M}_n(\mathbb{K})$, soit b un vecteur de $\mathbb{K}^n \setminus \{0\}$ et soit f le polynôme minimal de la suite $(A^i \cdot b)_{i \geq 0}$.
Le but de l'exercice est d'estimer la probabilité $\mathcal{P}$ que f coïncide avec le polynôme minimal de la suite $({}^t u \cdot A^i \cdot b)_{i \geq 0}$ lorsque u est un vecteur de $\mathbb{K}^n$ dont les coordonnées sont choisies aléatoirement au hasard dans un sous-ensemble fini U de $\mathbb{K}$.

1. Montrer qu'il existe une application $\psi : \mathbb{K}^n \to \mathbb{A} = \mathbb{K}[X]/(f)$, $\mathbb{K}$-linéaire et surjective, telle que pour tout $u \in \mathbb{K}^n$ on ait

   $$f \text{ est le polynôme minimal de } ({}^t u \cdot A^i \cdot b)_{i \geq 0} \iff \psi(u) \text{ est inversible dans } \mathbb{A}.$$

   [Indication : l'application $\phi : \mathbb{K}^n \to \mathbb{K}^{\mathbb{N}}$ définie par $\phi(u) = ({}^t u \cdot A^i \cdot b)_{i \geq 0}$ induit une application linéaire surjective $\mathbb{K}^n \to M_f$, où M_f est l'espace vectoriel des suites de $\mathbb{K}^{\mathbb{N}}$ admettant f comme polynôme annulateur. Par ailleurs, $\mathbb{A}$ et M_f sont isomorphes en tant qu'espaces vectoriels.]
2. Soit d le degré de f. Montrer qu'il existe un polynôme non identiquement nul $R \in \mathbb{K}[X_1, \ldots, X_n]$ de degré total au plus d tel que pour tout vecteur $u = (u_1, \ldots, u_n) \in \mathbb{K}^n$ on ait

   $$\psi(u) \text{ est inversible dans } \mathbb{A} \text{ si et seulement si } R(u_1, \ldots, u_n) \neq 0.$$

[Indication : utiliser un résultant.]

3. Montrer que R admet au plus $d \cdot \mathrm{card}(\mathrm{U})^{n-1}$ racines dans U^n. En déduire que la probabilité qu'un élément de U^n dont les coordonnées sont choisies aléatoirement au hasard dans U soit racine de R est bornée par $d/\mathrm{card}(\mathrm{U})$.
4. Conclure que la probabilité $\mathcal{P}$ vérifie

$$\mathcal{P} \geq 1 - \frac{d}{\mathrm{card}(\mathrm{U})}.$$

■

Notes

L'algorithme de Wiedemann est décrit dans son article de 1986 [Wie86], qui contient également l'algorithme esquissé en Section 9.4. L'algorithme en Section 9.5 est dû à Kaltofen et Saunders [KS91]. Des versions « par blocs » de l'algorithme de Wiedemann ont été proposées par Coppersmith [Cop94], Villard [Vil97], Thomé [Tho02], et Turner [Tur06]. D'autres généralisations se trouvent dans les travaux de Giesbrecht, Lobo et Saunders [Gie01 ; GLS98], et de Mulders [Mul04].

Les questions (2) et (3) de l'Exercice 9.1 forment le cœur du « lemme de Schwartz–Zippel » [DL78 ; Sch80b ; Zip79]. Nous en verrons une preuve en page 410 du Chapitre 22 sur la factorisation des polynômes à plusieurs variables.

Bibliographie

Cop94 Coppersmith, Don (1994). « Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm ». In : *Mathematics of Computation*, vol. 62, n°205, p. 333–350.

DL78 DeMillo, Richard A. et Richard J. Lipton (1978). « A probabilistic remark on algebraic program testing ». In : *Information Processing Letters*, vol. 7, n°4, p. 193–195.

Gie01 Giesbrecht, Mark (2001). « Fast computation of the Smith form of a sparse integer matrix ». In : *Computational Complexity*, vol. 10, n°1, p. 41–69.

GLS98 Giesbrecht, M., A. Lobo et B. D. Saunders (1998). « Certifying inconsistency of sparse linear systems ». In : *ISSAC'98 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 113–119.

KS91 Kaltofen, Erich et B. David Saunders (1991). « On Wiedemann's method of solving sparse linear systems ». In : *Applied algebra, algebraic algorithms and error-correcting codes (New Orleans, LA, 1991)*. Vol. 539. Lecture Notes in Computer Science. Springer-Verlag, p. 29–38.

Mul04 Mulders, Thom (2004). « Certified sparse linear system solving ». In : *Journal of Symbolic Computation*, vol. 38, n°5, p. 1343–1373.

Sch80b Schwartz, J. T. (1980). « Fast probabilistic algorithms for verification of polynomial identities ». In : *Journal of the Association for Computing Machinery*, vol. 27, n°4, p. 701–717.

Tho02 Thomé, E. (2002). « Subquadratic computation of vector generating polynomials and improvement of the block Wiedemann algorithm ». In : *Journal of Symbolic Computation*, vol. 33, n°5, p. 757–775.

Tur06 Turner, William J. (2006). « A block Wiedemann rank algorithm ». In : *ISSAC'06 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 332–339.

Vil97 Villard, G. (1997). « Further analysis of Coppersmith's block Wiedemann algorithm for the solution of sparse linear systems ». In : *ISSAC'97 : International Symposium on Symbolic and Algebraic Computation*. Kihei, Maui, Hawaii, United States : ACM Press, p. 32–39.

Wie86 Wiedemann, D. (1986). « Solving sparse linear equations over finite fields ». In : *IEEE Transactions on Information Theory*, vol. IT-32, p. 54–62.

Zip79 Zippel, Richard (1979). « Probabilistic algorithms for sparse polynomials ». In : *Symbolic and Algebraic Computation. Eurosam '79, An International Symposium on Symbolic and Algebraic Manipulation, Marseille, France, June 1979*. Vol. 72. Lecture Notes in Computer Science. Springer-Verlag, p. 216–226.

10. Algèbre linéaire structurée

Résumé

Les matrices possédant une structure spéciale sont omniprésentes en calcul formel. Ce sont des matrices dont les éléments jouissent d'une certaine répétition, ou satisfont à certaines relations. Plus formellement, une matrice $n \times n$ structurée est typiquement définie par $O(n)$ éléments, au lieu de n^2 pour une matrice dense, et peut être multipliée par un vecteur en $\tilde{O}(n)$ opérations arithmétiques, au lieu de $O(n^2)$ opérations pour une matrice dense. Ce chapitre présente une algorithmique unifiée pour manipuler des matrices structurées denses, comme les matrices de Toeplitz, de Hankel, de Vandermonde et de Sylvester. Les calculs avec les matrices de ces classes sont liés aux calculs avec les polynômes, ce qui permet l'emploi des techniques de multiplication polynomiale rapide pour accélérer leur manipulation. Par exemple, on peut résoudre un système linéaire défini par une matrice structurée inversible en $\tilde{O}(n)$ opérations.

10.1 Introduction

Nous avons vu au Chapitre 8 qu'il est possible de manipuler les matrices denses de taille $n \times n$ à coefficients dans un corps $\mathbb{K}$ en $\mathsf{MM}(n) = O(n^\theta)$ opérations dans $\mathbb{K}$, où θ est un réel compris entre 2 et 3. Ici, par manipuler, on entend multiplier, inverser, calculer le déterminant, ou encore résoudre un système linéaire.

Dans certaines situations, les matrices qu'on est amené à manipuler présentent une structure que ces algorithmes généraux ne savent pas détecter et exploiter. Voici quelques exemples de matrices qui possèdent une structure.

Définition 10.1 Une matrice $A \in \mathcal{M}_n(\mathbb{K})$ est dite *de type Toeplitz* (ou simplement *de Toeplitz*) si elle est invariante le long des diagonales, c'est-à-dire si ses éléments $a_{i,j}$ vérifient $a_{i,j} = a_{i+k,j+k}$ pour tout k.

Une telle matrice est complètement définie par sa première ligne et sa première colonne.

Définition 10.2 Une matrice $A \in \mathcal{M}_n(\mathbb{K})$ est dite *de Hankel* si elle est invariante le long des anti-diagonales, c'est-à-dire si ses éléments $a_{i,j}$ vérifient $a_{i,j} = a_{i-k,j+k}$ pour tout k.

Une telle matrice est complètement définie par sa première ligne et sa dernière colonne.

Exemple 10.1 La matrice de la multiplication polynomiale en degré fixé (dans les bases canoniques) par un polynôme fixé est de Toeplitz. Par exemple, la multiplication d'un polynôme de degré au plus 2 par le polynôme fixé $a_0 + a_1X + a_2X^2$ de $\mathbb{K}[X]$ se traduit matriciellement par l'égalité

$$\begin{pmatrix} a_0 & 0 & 0 \\ a_1 & a_0 & 0 \\ a_2 & a_1 & a_0 \\ 0 & a_2 & a_1 \\ 0 & 0 & a_2 \end{pmatrix} \times \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} a_0b_0 \\ a_0b_1 + a_1b_0 \\ a_0b_2 + a_1b_1 + a_2b_0 \\ a_1b_2 + a_2b_1 \\ a_2b_2 \end{pmatrix}.$$

Dans cet exemple, la matrice de Toeplitz est d'une forme particulière, appelée *bande*. En fait, toute matrice de Toeplitz peut être vue comme sous-matrice d'une matrice de Toeplitz bande, et cela entraîne le résultat suivant.

Lemme 10.1 Le produit d'une matrice de Toeplitz (ou de Hankel) de $\mathcal{M}_n(\mathbb{K})$ par un vecteur de $\mathbb{K}^n$ peut s'effectuer en $O(M(n))$ opérations.

Démonstration. Pour tout $0 \le i \le n-1$, l'élément c_i du produit matrice-vecteur

$$\begin{pmatrix} a_{n-1} & \cdots & a_0 \\ \vdots & \ddots & \vdots \\ a_{2n-2} & \cdots & a_{n-1} \end{pmatrix} \times \begin{pmatrix} b_0 \\ \vdots \\ b_{n-1} \end{pmatrix} = \begin{pmatrix} c_0 \\ \vdots \\ c_{n-1} \end{pmatrix}$$

est le coefficient de X^{n-1+i} dans le produit de polynômes

$$(a_0 + \cdots + a_{2n-2}X^{2n-2}) \times (b_0 + \cdots + b_{n-1}X^{n-1}). \tag{10.1}$$

La preuve est tout à fait similaire pour une matrice de Hankel. ∎

R La preuve précédente montre que $2\mathsf{M}(n)+\mathrm{O}(n)$ opérations suffisent pour multiplier une matrice de Toeplitz ou de Hankel de taille n par un vecteur. Cette borne peut être réduite à $\mathsf{M}(n)+\mathrm{O}(n)$, en observant que l'extraction de la *partie médiane* du produit (10.1) est suffisante, et en employant des algorithmes pour le produit médian, évoqués au Chapitre 12.

Exemple 10.2 Une matrice de Sylvester est la concaténation de deux matrices Toeplitz bande, cf. 6.2.

Définition 10.3 Une matrice $\mathrm{A} = (a_{i,j})_{i,j=0}^{n-1}$ de $\mathcal{M}_n(\mathbb{K})$ est dite *de Vandermonde* si ses éléments s'écrivent $a_{i,j} = a_i^j$ pour $a_0, \ldots, a_{n-1} \in \mathbb{K}$.

Définition 10.4 Une matrice $\mathrm{A} = (a_{i,j})_{i,j=0}^{n-1}$ de $\mathcal{M}_n(\mathbb{K})$ est dite *de Cauchy* si ses éléments s'écrivent $a_{i,j} = 1/(a_i - b_j)$ pour $a_i, b_j \in \mathbb{K}$ avec $a_i \neq b_j$ pour tous i, j.

Il y a un certain nombre de points en commun entre ces exemples : la matrice est représentable par $\mathrm{O}(n)$ éléments ; le produit matrice-vecteur peut s'effectuer plus rapidement que dans le cas générique, en $\mathrm{O}(\mathsf{M}(n)\log n)$ opérations au lieu de $\mathrm{O}(n^2)$; le produit par une matrice quelconque peut s'effectuer en complexité $\mathrm{O}(n\,\mathsf{M}(n)\log n)$ — quasi-optimale en la taille de la sortie pour une multiplication polynomiale à base de FFT. En effet, dans chaque cas, le produit matrice-vecteur $\mathrm{A}v$ admet une interprétation analytique :

— multiplication polynomiale dans les cas Toeplitz, Hankel et Sylvester ;
— évaluation polynomiale multipoint dans le cas d'une matrice de Vandermonde (Chapitre 5) ;
— évaluation multipoint de fractions rationnelles de la forme $\sum_{j=0}^{n-1} c_j/(\mathrm{X} - b_j)$ dans le cas d'une matrice de Cauchy.

Exercice 10.1 Montrer que le produit d'une matrice de Cauchy par un vecteur peut s'effectuer en $\mathrm{O}(\mathsf{M}(n)\log n)$ opérations dans $\mathbb{K}$. ∎

Au vu de ces exemples, on pourrait donc être tenté de définir comme *structurée* une matrice telle que son produit par un vecteur peut s'effectuer en $\tilde{\mathrm{O}}(n)$ opérations. La question qui se pose naturellement est : peut-on exploiter cette définition de la structure de A pour résoudre le système $\mathrm{A}x = b$? Un premier constat positif est que, dans chacun des exemples précédents, la résolution admet aussi une interprétation analytique :

i. devinette de récurrences à coefficients constants, si A est de Toeplitz ou de Hankel ;
ii. interpolation polynomiale, si A est de Vandermonde ;
iii. interpolation de fractions rationnelles, si A est de Cauchy.

En effet, si la suite (a_n) d'éléments de $\mathbb{K}$ vérifie une récurrence (inconnue) à coefficients constants de la forme

$$a_{n+d} = p_{d-1}a_{n+d-1} + \cdots + p_0 a_n, \qquad n \geq 0,$$

alors trouver les coefficients p_i de cette récurrence revient à résoudre le système de Hankel

$$\begin{pmatrix} a_0 & a_1 & \cdots & a_{d-1} \\ a_1 & a_2 & \iddots & a_d \\ \vdots & \iddots & \iddots & \vdots \\ a_{d-1} & a_d & \cdots & a_{2d-2} \end{pmatrix} \times \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_{d-1} \end{pmatrix} = \begin{pmatrix} a_d \\ a_{d+1} \\ \vdots \\ a_{2d-1} \end{pmatrix}.$$

Un autre point encourageant est que, pour ces trois opérations, on dispose d'algorithmes de complexité quasi-optimale $\mathrm{O}(\mathrm{M}(n)\log n)$, présentés au Chapitre 5 pour (ii), et au Chapitre 7 pour (i) et (iii).

Il y a cependant quelques points négatifs : un premier est que si A est une matrice inversible telle que l'application linéaire $v \mapsto \mathrm{A}v$ se calcule en L opérations, cela n'implique pas l'existence d'un algorithme de complexité O(L) pour l'application $v \mapsto \mathrm{A}^{-1}v$. En d'autres termes, on manque d'un *principe d'inversion* analogue au *principe de transposition* présenté au Chapitre 12. Par exemple, pour les matrices creuses, le meilleur algorithme de résolution, dû à Wiedemann, est de complexité quadratique en n (Chapitre 9). Un second point négatif est que les classes vues plus haut ne sont pas stables par inversion : l'inverse d'une matrice de Toeplitz (resp. de Vandermonde) n'est pas de Toeplitz (resp. de Vandermonde).

On a donc besoin de rajouter des hypothèses dans la définition de la *bonne notion* de *matrice structurée*. Cette définition exploite la généralisation du caractère invariant par diagonale d'une matrice de Toeplitz, et vient de l'observation simple suivante : si A est de Toeplitz, alors la matrice

$$\phi(\mathrm{A}) = \mathrm{A} - (\text{A décalée de 1 vers le bas et de 1 vers la droite})$$

admet une sous-matrice carrée nulle d'ordre un de moins ; elle est donc de rang au plus 2 (on convient qu'on remplit avec des zéros les premières ligne et colonne de la matrice décalée). On dit alors que ϕ est un *opérateur de déplacement* et que le *rang de déplacement par rapport à* ϕ de A est au plus 2. On peut donc représenter $\phi(\mathrm{A})$ sous forme compacte, comme un produit $\mathrm{G}\cdot{}^t\mathrm{H}$, avec G et H des matrices rectangulaires de taille $n \times 2$.

Exemple 10.3 Si A est la matrice 3×3

$$\mathrm{A} = \begin{pmatrix} c & d & e \\ b & c & d \\ a & b & c \end{pmatrix},$$

avec $d \neq 0$, alors $\phi(\mathrm{A})$ s'écrit

$$\phi(\mathrm{A}) = \begin{pmatrix} c & d & e \\ b & 0 & 0 \\ a & 0 & 0 \end{pmatrix} = \begin{pmatrix} c & d \\ b & 0 \\ a & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & e/d \end{pmatrix}.$$

Définition 10.5 Les matrices (G, H) de $\mathcal{M}_{n,2}(\mathbb{K})$ telles que $\phi(A) = G \cdot {}^tH$ sont appelées *générateurs de déplacement* pour la matrice de Toeplitz A.

Ces définitions s'étendent à une matrice quelconque et conduisent au concept de matrice *quasi-Toeplitz*.

Définition 10.6 On appelle plus généralement *opérateur de déplacement* ϕ_+ l'application $A \mapsto A - Z \cdot A \cdot {}^tZ$, où A est une matrice quelconque et Z la matrice définie par

$$Z = \begin{pmatrix} 0 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ \vdots & \ddots & \dots & \vdots \\ 0 & \cdots & 1 & 0 \end{pmatrix}.$$

ZA est la matrice A décalée de 1 ligne vers le bas, et $A \cdot {}^tZ$ est la matrice A décalée de 1 colonne vers la droite, si bien que l'on retrouve bien la définition précédente pour matrice de Toeplitz. Le *rang de déplacement* de A est comme auparavant l'entier $\alpha_+(A) = \text{rang}(\phi_+(A))$. On appelle également *générateur de déplacement* pour l'opérateur ϕ_+ un couple (G, H) de matrices $n \times \alpha$ vérifiant l'égalité $\phi_+(A) = G \cdot {}^tH$. Si $\alpha_+(A) \ll n$, on dit que A est *quasi-Toeplitz*. L'entier $\alpha_+(A)$ est appelé *longueur du générateur* (G, H).

R L'indice + est introduit pour faire pendant à un indice – utilisé ultérieurement (cf Définition 10.8).

Intuitivement, le rang de déplacement mesure à quel point la matrice A est loin d'être de Toeplitz ou d'une concaténation de Toeplitz.

Exemple 10.4 Les matrices de Toeplitz et Sylvester sont quasi-Toeplitz, de rang de déplacement 2.

Une autre extension permet de définir des matrices *quasi-Vandermonde* et *quasi-Cauchy*, mais pour des opérateurs de déplacement différents :

— $\mathbf{V_a} \mapsto \mathbf{V_a} - \text{Diag}_\mathbf{a} \cdot \mathbf{V_a} \cdot {}^tZ$ pour le cas Vandermonde ;
— $\mathbf{C_{a,b}} \mapsto \mathbf{C_{a,b}} - \text{Diag}_\mathbf{a}^{-1} \cdot \mathbf{C_{a,b}} \cdot \text{Diag}_\mathbf{b}$ pour le cas Cauchy.

Ici, pour deux vecteurs $\mathbf{a} = (a_0, \dots, a_{n-1})$ et $\mathbf{b} = (b_0, \dots, b_{n-1})$, on désigne par $\mathbf{V_a}$ la matrice de Vandermonde $(a_i^j)_{i,j=0}^{n-1}$, par $\text{Diag}_\mathbf{a}$ la matrice diagonale, dont la diagonale porte le vecteur $\mathbf{a}$, et par $\mathbf{C_{a,b}}$ la matrice de Cauchy $(1/(a_i - b_j))_{i,j=0}^{n-1}$.

Dans tous ces cas, l'application $\phi_{M,N}$ est définie par $\phi_{M,N}(A) = A - MAN$, avec M et N bien choisies (en fonction de la structure visée), et le rang de déplacement de A est défini comme étant le rang de la matrice $\phi_{M,N}(A)$.

Exercice 10.2 Calculer les rangs de déplacement des matrices de Vandermonde et de Cauchy, pour les opérateurs de déplacement définis ci-dessus. ■

Résultat principal

L'idée-clé derrière les algorithmes rapides pour les matrices structurées de rang de déplacement α est l'utilisation des générateurs de déplacement comme structure de données compacte, de taille $O(\alpha n)$, donc proportionnelle à la taille n de la matrice, dans le cas où $\alpha \ll n$. Le résultat principal de ce chapitre est le suivant.

Théorème 10.2 (Cadre quasi-Toeplitz, quasi-Vandermonde, quasi-Cauchy). Soit α le rang de déplacement adéquat, A une matrice de $\mathcal{M}_n(\mathbb{K})$ donnée par des générateurs de déplacement de taille $n \times \alpha$, et soit b un vecteur de $\mathbb{K}^n$. Alors, il est possible de :

1. calculer le déterminant de A ;
2. calculer le rang de A ;
3. calculer une solution du système $Ax = b$, ou prouver qu'il n'y en a aucune

en $\tilde{O}(\alpha^2 n)$ opérations dans $\mathbb{K}$. Plus exactement, cette complexité s'exprime en termes de la fonction de multiplication polynomiale, et vaut :

1. $O(\alpha^2 M(n) \log n)$ dans le cas quasi-Toeplitz ;
2. $O(\alpha^2 M(n) \log^2 n)$ dans les cas quasi-Vandermonde et quasi-Cauchy.

Ce théorème a pour conséquence une *algorithmique unifiée* pour résoudre en complexité quasi-optimale différents problèmes sur les polynômes et les séries.

Corollaire 10.3 On peut calculer en $O(M(n) \log n)$ opérations arithmétiques :

1. le pgcd étendu et le résultant de deux polynômes de degré borné par n ;
2. un approximant de Padé de type (n, n) d'une série tronquée à précision $2n$.

Démonstration. Le résultant de deux polynômes $A, B \in \mathbb{K}[X]$ s'obtient comme le déterminant de leur matrice de Sylvester $\mathrm{Syl}(A, B)$, qui a un rang de déplacement au plus 2. Le degré du pgcd $G = \mathrm{pgcd}(A, B)$ s'obtient par un calcul de rang de la matrice de Sylvester : $\deg(G) = \deg(A) + \deg(B) - \mathrm{rang}(\mathrm{Syl}(A, B))$. Une fois connu le degré du pgcd, la relation de Bézout $UA + VB = G$, avec les contraintes $\deg(U) < \deg(B) - \deg(G)$ et $\deg(V) < \deg(A) - \deg(G)$, se traduit en un système linéaire en les coefficients de U, V et G, dont la matrice est une concaténation de trois matrices de Toeplitz, donc une matrice quasi-Toeplitz de rang de déplacement au plus 3. Des considérations similaires s'appliquent pour le calcul d'approximants de Padé. ■

Corollaire 10.4 Étant données n series $f_1, \ldots, f_n$ de $\mathbb{K}[[X]]$ connues à précision $\sigma = \sum_i (d_i + 1) - 1$, il est possible d'en calculer un approximant de Padé–Hermite $(p_1, \ldots, p_n)$ de type $(d_1, \ldots, d_n)$ en $O(n^2 M(\sigma) \log \sigma)$ opérations dans $\mathbb{K}$.

Démonstration. Il s'agit d'un problème linéaire en les coefficients de l'approximant cherché. La matrice est quasi-Toeplitz, de rang de déplacement borné par n. ■

Dans la suite, nous allons esquisser les grandes lignes de la preuve du Théorème 10.2 dans le cas particulier où la matrice A est inversible et « suffisamment générique » (tous les mineurs non nuls), et uniquement dans le cas quasi-Toeplitz.

10.2 Le cas quasi-Toeplitz

Dans le reste du chapitre, nous allons nous concentrer uniquement sur le cas quasi-Toeplitz, car il contient les principales idées algorithmiques, et il couvre beaucoup d'applications ; les cas quasi-Vandermonde et quasi-Cauchy s'y ramènent et sont plus techniques.

Nous allons montrer que la notion de matrice quasi-Toeplitz est une bonne notion de structure, car elle répond aux propriétés suivantes :

(P1) on peut effectuer quasi-optimalement le produit d'une matrice quasi-Toeplitz par un vecteur ;
(P2) la somme et le produit de deux matrices quasi-Toeplitz restent quasi-Toeplitz ;
(P3) l'inverse aussi.

Or, ces propriétés forment le minimum nécessaire pour concevoir un algorithme de type *inversion de Strassen* (page 173) dans la représentation compacte par générateurs de déplacement.

Produit matrice-vecteur, cas quasi-Toeplitz

Pour montrer la propriété (P1) ci-dessus, le point clé est le suivant.

Proposition 10.5 — Formule ΣLU. L'opérateur $\phi_+ : A \mapsto A - Z \cdot A \cdot {}^tZ$ est inversible. Plus exactement, on a la formule suivante, appelée *formule* ΣLU :

$$A - Z \cdot A \cdot {}^tZ == G \cdot {}^tH \quad \text{si et seulement si} \quad A = \sum_{i=1}^{\alpha} L(x_i) \cdot U(y_i),$$

où les x_i (resp. y_i) sont les colonnes du générateur G (resp. H), et où, pour un vecteur colonne $v = {}^t[v_0 \cdots v_{n-1}]$, on note $L(v)$ la matrice de Toeplitz triangulaire inférieure

$$\begin{pmatrix} v_0 & 0 & \dots & 0 \\ v_1 & v_0 & \dots & 0 \\ \vdots & \ddots & \dots & \vdots \\ v_{n-1} & \cdots & v_1 & v_0 \end{pmatrix}$$

et $U(v)$ la matrice de Toeplitz triangulaire supérieure ${}^tL(v)$.

Démonstration. Par linéarité, il suffit de traiter le cas $\alpha = 1$. Si $C = L(a)U(b)$, alors un calcul immédiat montre que $c_{i,j} = a_i b_j + a_{i-1} b_{j-1} + \cdots$, et donc $c_{i,j} - c_{i-1,j-1} = a_i b_j$ et $\phi_+(C) = (a_i b_j)_{i,j=0}^{n-1} = a \cdot {}^t b$.

L'implication inverse découle de l'injectivité de ϕ_+ et est laissée en exercice. ■

La Proposition 10.5 permet de donner une définition équivalente pour le rang de déplacement.

Définition 10.7 Le nombre $\alpha_+(A)$ est le plus petit entier α tel qu'il existe une

décomposition de la forme

$$A = \sum_{i=1}^{\alpha} L_i U_i$$

pour des matrices L_i de Toeplitz inférieures, et U_i de Toeplitz supérieures.

Exemple 10.5 Si A est de Toeplitz, alors en notant A_{inf} sa partie inférieure et A_{ssup} sa partie strictement supérieure, on a que A admet la décomposition $A = A_{inf} \cdot I_n + I_n \cdot A_{ssup}$, donc $\alpha_+(A) \leq 2$

La Définition 10.7 permet de prouver la propriété (P1) en page 201.

Corollaire 10.6 Si A est donnée en représentation compacte par une paire de générateurs de déplacement (G, H) de taille $n \times \alpha$, alors le produit matrice-vecteur Av peut s'effectuer en $O(\alpha M(n))$ opérations arithmétiques.

Démonstration. Le produit Av s'écrit comme la somme de α termes, tous de la forme $L(x_i)(U(y_i)v)$. Or, chacun de ces termes peut être calculé en $O(M(n))$ opérations, grâce au Lemme 10.1. ■

Addition et produit en représentation compacte par générateurs

Pour justifier la propriété (P2) en page 201, on va montrer plus, à savoir que la représentation par générateurs permet un calcul efficace (en temps quasi-linéaire) des opérations d'additions et de multiplication de deux matrices quasi-Toeplitz.

Proposition 10.7 — Opérations matricielles en représentation compacte. Soient (T, U) générateur de déplacement de longueur α pour A, et (G, H) générateur de longueur β pour B. Alors

1. $([T|G], [U|H])$ sont des générateurs de déplacement pour $A + B$ de longueur $\alpha + \beta$;
2. $([T|W|a], [V|H|-b])$ sont des générateurs de déplacement pour AB, de longueur $\alpha + \beta + 1$,

où $V := {}^tB \cdot U$, $W := Z \cdot A \cdot {}^tZ \cdot G$, et où le vecteur a (resp. b) est la dernière colonne de $Z \cdot A$ (resp. de $Z \cdot {}^tB$).

La preuve, basée sur une vérification immédiate, est laissée en exercice.

Corollaire 10.8 En représentation compacte par générateurs de déplacement, de longueurs au plus α pour A et B, on peut calculer

1. la somme $A + B$ en $O(\alpha n)$ opérations dans $\mathbb{K}$;
2. le produit AB en $Mul(n, \alpha) := O(\alpha^2 M(n))$ opérations dans $\mathbb{K}$.

Démonstration. Le seul point non trivial est le calcul des matrices V et W et des vecteurs a et b. Tout repose sur la formule ΣLU. Si B s'écrit $\sum_{i=1}^{\beta} L(x_i)U(y_i)$, alors

sa transposée s'écrit ${}^t\mathrm{B} = \sum_{i=1}^{\beta} \mathrm{L}(y_i)\mathrm{U}(x_i)$, et le calcul de $\mathrm{V} = {}^t\mathrm{B}\cdot\mathrm{U}$ se ramène à $2\alpha\beta$ multiplications polynomiales, chacune pouvant s'effectuer en $\mathrm{O}(\mathsf{M}(n))$. Le même raisonnement s'applique au calcul de W. Enfin, le calcul de a revient à multiplier A par la colonne ${}^t[0,\ldots,0,1]$ et une observation similaire pour b permet de conclure. ■

Inversion en représentation compacte par générateurs

Afin de prouver la propriété (P3) en page 201, il est commode d'introduire un *opérateur dual de déplacement*.

Définition 10.8 L'opérateur de ϕ_--déplacement d'une matrice A de $\mathcal{M}_n(\mathbb{K})$ est défini par l'égalité

$$\begin{aligned}\phi_-(\mathrm{A}) &= \mathrm{A} - {}^t\mathrm{Z}\cdot\mathrm{A}\cdot\mathrm{Z}\\ &= \mathrm{A} - (\mathrm{A}\text{ décalée de 1 vers le haut et vers la gauche}).\end{aligned}$$

Le rang de déplacement α_- est défini par la formule

$$\alpha_-(\mathrm{A}) = \mathrm{rang}(\phi_-(\mathrm{A})).$$

On peut montrer (de la même façon que pour ϕ_+) que l'on dispose d'une caractérisation équivalente pour ϕ_-, et que ϕ_+ et ϕ_- sont liés.

Lemme 10.9

1. $\alpha_-(\mathrm{A})$ est le plus petit entier α tel que A puisse s'écrire sous la forme $\sum_{i=1}^{\alpha}\mathrm{U}_i\mathrm{L}_i$, avec les L_i de Toeplitz inférieures, et les U_i de Toeplitz supérieures.
2. On a la *formule* ΣUL *:*

$$\mathrm{A} - {}^t\mathrm{Z}\cdot\mathrm{A}\cdot\mathrm{Z} = \sum_{i=1}^{\alpha} x_i\cdot{}^t y_i \quad\text{si et seulement si}\quad \mathrm{A} = \sum_{i=1}^{\alpha}\mathrm{U}(\mathrm{rev}(x_i))\cdot\mathrm{L}(\mathrm{rev}(y_i)).$$

   Ici, pour $v = {}^t[v_0,\ldots,v_{n-1}]$, on note $\mathrm{rev}(v) = {}^t[v_{n-1},\ldots,v_0]$.
3. Une matrice de Toeplitz pour ϕ_+ est une matrice de Toeplitz pour ϕ_-.

La preuve de ce résultat découle de la proposition suivante.

Proposition 10.10 — Conversion ΣLU ↔ ΣUL. Pour toute matrice A, l'inégalité $|\alpha_+(\mathrm{A}) - \alpha_-(\mathrm{A})| \le 2$ est satisfaite. De plus, on peut effectuer les conversions d'une représentation $\sum \mathrm{LU}$ à une représentation $\sum \mathrm{UL}$, et inversement, en $\mathrm{O}(\alpha\,\mathsf{M}(n))$ opérations dans $\mathbb{K}$.

Démonstration. Il suffit de prouver l'identité

$$\mathrm{L}(x)\cdot\mathrm{U}(y) = \mathrm{I}_n\cdot\mathrm{L}(y') + \mathrm{U}(x')\cdot\mathrm{I}_n - \mathrm{U}(x'')\cdot\mathrm{L}(y''),$$

où $x'' = \mathrm{Z}\cdot\mathrm{rev}(x)$, $y'' = \mathrm{Z}\cdot\mathrm{rev}(y)$, $y' = \mathrm{rev}({}^t\mathrm{U}(y)\cdot{}^t\mathrm{L}(x)\cdot f)$, $x' = \mathrm{rev}(\mathrm{L}(x)\cdot\mathrm{U}(y)\cdot f)$, et $f = {}^t[0,\ldots,0,1]$. ■

Entrée Une matrice « générique » $A \in \mathcal{M}_n(\mathbb{K})$, de taille $n = 2^k$, représentée par des générateurs (pour l'opérateur de déplacement ϕ_+).
Sortie Son inverse A^{-1}, représentée par des générateurs (pour l'opérateur de déplacement ϕ_-).

1. Si $n = 1$, renvoyer A^{-1}.
2. Calculer des ϕ_+-générateurs pour $a, b, c, d \in \mathcal{M}_{n/2}(\mathbb{K})$, où $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$.
3. Calculer récursivement des ϕ_--générateurs pour $e := a^{-1}$.
4. Calculer des ϕ_+-générateurs pour $S := d - ceb$.
5. Calculer récursivement des ϕ_--générateurs pour $t := S^{-1}$.
6. Renvoyer des ϕ_--générateurs de $A^{-1} = \begin{pmatrix} x & y \\ z & t \end{pmatrix}$ via les formules de Strassen $y := -ebt$, $z := -tce$ et $x := e + ebtce$.

Algorithme 10.1 – Algorithme de type Strassen pour inverser une matrice quasi-Toeplitz.

Théorème 10.11 Soit $A \in \mathcal{M}_n(\mathbb{K})$ une matrice quasi-Toeplitz inversible. Alors son inverse est aussi quasi-Toeplitz et $\alpha_+(A^{-1}) = \alpha_-(A)$.

Démonstration. On a la suite d'égalités

$$\begin{aligned}\alpha_-(A) &= \operatorname{rang}(A - {}^tZ \cdot A \cdot Z) = \operatorname{rang}(I_n - A^{-1} \cdot {}^tZ \cdot A \cdot Z) = \operatorname{rang}(I_n - Z \cdot A^{-1} \cdot {}^tZ \cdot A) \\ &= \operatorname{rang}(A^{-1} - Z \cdot A^{-1} \cdot {}^tZ) = \alpha_+(A^{-1}),\end{aligned}$$

dans laquelle nous avons utilisé un fait d'algèbre linéaire élémentaire : si A, B sont deux matrices quelconques, alors $\operatorname{rang}(I_n - A \cdot B) = \operatorname{rang}(I_n - B \cdot A)$. ∎

Résolution rapide de systèmes quasi-Toeplitz

Le Théorème 10.11 ne nous dit pas *comment* calculer des générateurs pour A^{-1}. Cela est fait dans le dernier résultat de ce chapitre.

Théorème 10.12 Soit $A \in \mathcal{M}_n(\mathbb{K})$ une matrice quasi-Toeplitz inversible, de rang de déplacement α, donnée par des générateurs G et H de longueur α. On peut calculer des générateurs de longueur α pour A^{-1} en $O(\alpha^2 M(n) \log n)$ opérations dans $\mathbb{K}$.

À partir de cette représentation de l'inverse et d'un vecteur $b \in \mathbb{K}^n$, le système $Ax = b$ peut se résoudre pour un supplément de $O(\alpha M(n))$ opérations dans $\mathbb{K}$.

Démonstration. L'idée est d'adapter au cas structuré l'algorithme d'inversion de Strassen présenté au Chapitre 8 (page 173). La différence réside dans le choix de la structure

de données utilisée pour représenter et calculer avec les matrices quasi-Toeplitz. Ici, on utilise comme structure de données compacte les générateurs de déplacement pour les opérateurs ϕ_+ et ϕ_- (ou, de façon équivalente, les représentations ΣLU et ΣUL).

La méthode, de type « diviser pour régner », est résumée dans l'Algorithme 10.1. Sa correction est héritée de celle de l'algorithme d'inversion de Strassen ; l'hypothèse sur la généricité de A est utilisée pour garantir que tous les mineurs qu'on doit inverser au cours de l'algorithme sont bien inversibles.

Sa complexité arithmétique $\mathsf{C}(n)$ obéit à la récurrence $\mathsf{C}(1) = 1$ et, si $n \geq 2$,

$$\mathsf{C}(n) = 2\,\mathsf{C}(n/2) + \mathrm{O}(\mathsf{Mul}(n,\alpha)) + \mathrm{O}(\alpha^2 n + \alpha\,\mathsf{M}(n)),$$

où le coût en $\mathrm{O}(\alpha^2 n + \alpha\,\mathsf{M}(n))$ provient des conversions entre les représentations ϕ_+ et ϕ_- et des calculs de minimisation de longueurs de générateurs, et la notation $\mathsf{Mul}(n,\alpha)$ désigne la complexité du produit matriciel pour des matrices $n \times n$ données par générateurs de taille $n \times \alpha$. La preuve se conclut en utilisant l'estimation

$$\mathsf{Mul}(n,\alpha) = \mathrm{O}(\alpha^2\,\mathsf{M}(n)),$$

qui entraîne $\mathsf{C}(n) = \mathrm{O}(\alpha^2\,\mathsf{M}(n)\log n)$. ■

Exercices

Exercice 10.3 Soit $\mathrm{A} \in \mathcal{M}_n(\mathbb{K})$ une matrice quasi-Toeplitz de rang de déplacement $\alpha \ll n$, représentée de façon compacte par des générateurs (G,H) de taille $n \times \alpha$, c'est-à-dire telle que $\phi_+(\mathrm{A}) = \mathrm{G} \cdot {}^t\mathrm{H}$.

1. Soient $v_1, \ldots, v_\alpha$ des vecteurs quelconques de $\mathbb{K}^n$. Montrer que le calcul de tous les produits $\mathrm{A} \cdot v_\ell$, $1 \leq \ell \leq \alpha$, se ramène, grâce à la formule ΣLU, au problème suivant :
   (P) Étant donnés des polynômes $\mathrm{G}_j, \mathrm{H}_j, \mathrm{V}_j \in \mathbb{K}[\mathrm{X}]$ $(j \leq \alpha)$ de degrés au plus $n-1$,
   $$\text{calculer } \mathrm{A}_\ell = \sum_{j=1}^{\alpha} \mathrm{G}_j\,(\mathrm{H}_j \mathrm{V}_\ell \bmod \mathrm{X}^n),\ 1 \leq \ell \leq \alpha.$$
2. Donner un premier algorithme qui résout le problème **(P)** et estimer sa complexité.
3. Montrer que le problème **(P)** admet la reformulation matricielle suivante :
   (MP) Étant données des matrices polynomiales $\mathbf{G}, \mathbf{V}$ dans $\mathcal{M}_{\alpha\times 1}(\mathbb{K}[\mathrm{X}])$ et $\mathbf{H}$ dans $\mathcal{M}_{1\times\alpha}(\mathbb{K}[\mathrm{X}])$, toutes de degrés au plus $n-1$, calculer $(\mathbf{VH} \bmod \mathrm{X}^n)\mathbf{G}$.
4. Soient $\mathbf{A}, \mathbf{B}$ et $\mathbf{C}$ des matrices polynomiales de tailles $(n \times p)$, $(p \times n)$ et $(n \times p)$ et de degré au plus d. Montrer que le produit $\mathbf{ABC}$ peut être calculé en $\mathrm{O}(\frac{n}{p}\,\mathsf{MM}(p,d))$ opérations dans $\mathbb{K}$.
5. Proposer un algorithme de type « diviser pour régner » (par rapport à n) résolvant le problème **(MP)** en complexité $\mathrm{O}(\frac{1}{\alpha}\,\mathsf{MM}(\alpha,n))$.
6. Conclure qu'on peut multiplier, dans la représentation par générateurs de déplacement, deux matrices quasi-Toeplitz de $\mathcal{M}_n(\mathbb{K})$ de rang de déplacement au plus α en $\mathrm{O}(\frac{1}{\alpha}\,\mathsf{MM}(\alpha,n))$ opérations dans $\mathbb{K}$.

■

Notes

Levinson [Lev47] a donné le premier algorithme de complexité quadratique pour la résolution d'un système linéaire dont la matrice est de Toeplitz, symétrique et définie positive. Trench [Tre64] a par la suite montré que tout l'inverse d'une telle matrice peut se calculer en la même complexité. Ces deux algorithmes sont décrits dans le livre de Golub et Van Loan [GV13, §4.7].

L'algorithme de Trench [Tre64] repose sur une formule, reprise et améliorée par Gohberg et Semencul [GS72], qui fournit une représentation explicite $\Sigma\mathrm{LU}$ à deux termes de l'inverse d'une matrice de Toeplitz A, les sommands étant construits à partir de la première ligne et de la première colonne de A^{-1}. Brent, Gustavson et Yun [BGY80 ; GY79] ont montré comment ramener le calcul des premières ligne et colonne de A^{-1} à un calcul d'approximation de Padé. Ils ont aussi mis en évidence les liens profonds entre la résolution des systèmes de Toeplitz, l'algorithme d'Euclide étendu et l'approximation de Padé, et ont proposé le premier algorithme pour la résolution d'un système défini par une matrice de Toeplitz inversible quelconque de taille n en $\mathrm{O}(\mathsf{M}(n)\log n)$ opérations arithmétiques.

Parallèlement, la notion de rang de déplacement a été introduite par Kailath, Kung et Morf dans [KKM79b], pour pouvoir résoudre un système quasi-Toeplitz inversible de rang de déplacement α en coût $\mathrm{O}(\alpha n^2)$ [KKM79a]. Des versions rapides, de complexité $\mathrm{O}(\alpha^d \mathsf{M}(n)\log n)$ ont été obtenues indépendamment par Bitmead et Anderson [BA80] (avec $d = 4$) et Morf [Mor80] (avec $d = 2$), sous des hypothèses de forte régularité sur l'entrée. Ces hypothèses ont été ensuite levées par Kaltofen [Kal94 ; Kal95b]. Les extensions au cas quasi-Cauchy sont dues à Cardinal [Car99], Pan et Zheng [PZ00]. Le cas quasi-Vandermonde est traité dans des articles de Pan [Pan90], et de Gohberg et Olshevsky [GO94]. La meilleure complexité asymptotique vis-à-vis simultanément de la taille de la matrice et de son rang de déplacement est due à Bostan, Jeannerod et Schost [BJS08], qui ont trouvé comment introduire du produit rapide de matrices denses dans l'algorithmique des matrices structurées, et ont réduit le coût à $\tilde{\mathrm{O}}(\alpha^{\theta-1} n)$ pour la résolution des systèmes quasi-Toeplitz, quasi-Vandermonde et quasi-Cauchy.

Une bonne référence sur les matrices structurées est le livre de Pan [Pan01].

Bibliographie

BA80 Bitmead, Robert R. et Brian D. O. Anderson (1980). « Asymptotically fast solution of Toeplitz and related systems of linear equations ». In : *Linear Algebra and its Applications*, vol. 34, p. 103–116.

BGY80 Brent, Richard P., Fred G. Gustavson et David Y. Y. Yun (1980). « Fast solution of Toeplitz systems of equations and computation of Padé approximants ». In : *Journal of Algorithms*, vol. 1, n°3, p. 259–295.

BJS08 Bostan, Alin, Claude-Pierre Jeannerod et Éric Schost (2008). « Solving structured linear systems with large displacement rank ». In : *Theoretical Computer Science*, vol. 407, n°1-3, p. 155–181.

Car99 Cardinal, Jean-Paul (1999). « On a property of Cauchy-like matrices ». In : *Comptes Rendus de l'Académie des Sciences. Série I. Mathématique*, vol. 328, n°11, p. 1089–1093.

GO94 Gohberg, I. et V. Olshevsky (1994). « Complexity of multiplication with vectors for structured matrices ». In : *Linear Algebra and its Applications*, vol. 202, p. 163–192.

GS72 Gohberg, I. C. et A. A. Semencul (1972). « On the inversion of finite Toeplitz matrices and their continuous analogues (in Russian) ». In : *Matematicheskie Issledovaniya*, vol. 7, n°2, p. 201–223.

GV13 Golub, Gene H. et Charles F. Van Loan (2013). *Matrix computations*. Fourth. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD.

GY79 Gustavson, Fred G. et David Y. Y. Yun (1979). « Fast algorithms for rational Hermite approximation and solution of Toeplitz systems ». In : *IEEE Transactions on Circuits and Systems*, vol. 26, n°9, p. 750–755.

Kal94 Kaltofen, Erich (1994). « Asymptotically fast solution of Toeplitz-like singular linear systems ». In : *ISSAC'94 : International Symposium on Symbolic and Algebraic Computation*. Oxford, United Kingdom : ACM Press, p. 297–304.

Kal95b — (1995). « Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems ». In : *Mathematics of Computation*, vol. 64, n°210, p. 777–806.

KKM79a Kailath, T., S. Y. Kung et M. Morf (1979). « Displacement ranks of a matrix ». In : *American Mathematical Society. Bulletin. New Series*, vol. 1, n°5, p. 769–773.

KKM79b Kailath, Thomas, Sun Yuan Kung et Martin Morf (1979). « Displacement ranks of matrices and linear equations ». In : *Journal of Mathematical Analysis and Applications*, vol. 68, n°2, p. 395–407.

Lev47 Levinson, Norman (1947). « The Wiener RMS (root mean square) error criterion in filter design and prediction ». In : *Journal of Mathematics and Physics*, vol. 25, p. 261–278.

Mor80 Morf, M. (1980). « Doubling algorithms for Toeplitz and related equations ». In : *IEEE Conference on Acoustics, Speech, and Signal Processing*, p. 954–959.

Pan01 Pan, Victor Y. (2001). *Structured matrices and polynomials*. Unified superfast algorithms. Birkhäuser Boston Inc.

Pan90 Pan, Victor (1990). « On computations with dense structured matrices ». In : *Mathematics of Computation*, vol. 55, n°191, p. 179–190.

PZ00 Pan, Victor Y. et Ailong Zheng (2000). « Superfast algorithms for Cauchy-like matrix computations and extensions ». In : *Linear Algebra and its Applications*, vol. 310, n°1-3, p. 83–108.

Tre64 Trench, William F. (1964). « An algorithm for the inversion of finite Toeplitz matrices ». In : *Journal of the Society for Industrial and Applied Mathematics*, vol. 12, p. 515–522.

11. Solutions rationnelles de systèmes linéaires à coefficients polynomiaux

Résumé

L'algorithmique des systèmes linéaires à coefficients des polynômes en une variable est très similaire à celle des fractions rationnelles. En outre, il est important de tirer parti du produit rapide de matrices.

On considère le système linéaire

$$A(X)Y(X) = B(X), \tag{11.1}$$

où A et B sont donnés et Y est inconnu. A est une matrice $n \times n$ de polynômes, régulière (de déterminant non nul) et B est un vecteur de polynômes. De manière équivalente, on peut voir A (ou B) comme un polynôme à coefficients des matrices (ou des vecteurs). Pour fixer les notations, on suppose $\deg A \leq d$ et $\deg B < d$.

On notera $\mathcal{M}_{m,k}(R)$ l'ensemble des matrices de taille $m \times k$ à coefficients dans R. On notera $\mathbb{K}[X]_d$ l'ensemble des polynômes de degré au plus d à coefficients dans le corps $\mathbb{K}$. La fonction M est telle que la multiplication dans $\mathbb{K}[X]_d$ coûte au plus $\mathsf{M}(d)$ opérations dans $\mathbb{K}$. L'exposant θ est tel que la multiplication de deux matrices $n \times n$ à coefficients dans $\mathbb{K}$ coûte $\mathsf{MM}(n) = O(n^\theta)$ opérations dans $\mathbb{K}$.

Nous aurons aussi besoin de produits de matrices de $\mathcal{M}_n(\mathbb{K}[X]_d)$. Dans le cas le plus général, ce produit est connu pour pouvoir être exécuté en $\mathsf{MM}(n,d) = O(n^\theta \mathsf{M}(d))$ opérations dans $\mathbb{K}$, borne que nous utiliserons dans les estimations de complexité. Lorsque le corps $\mathbb{K}$ contient suffisamment de points, ce coût descend par évaluation-interpolation à $O(n^\theta d + n^2 \mathsf{M}(d) \log d)$; dans les mêmes conditions, en choisissant des points en progression géométrique, cette complexité peut être encore abaissée à $O(n^\theta d + n^2 \mathsf{M}(d))$, voir page 106.

Entrée $A \in \mathcal{M}_n(\mathbb{K}[X]_d)$, $B \in \mathcal{M}_{n,1}(\mathbb{K}[X]_{d-1})$.
Sortie le vecteur de fractions rationnelles Y tel que $AY = B$.
1. Calculer le développement en série de $A^{-1}B$ à précision $2nd$.
2. Reconstruire les coefficients de Y par approximant de Padé.

Algorithme 11.1 – Résolution de $A(X)Y(X) = B(X)$.

11.1 Des séries aux solutions rationnelles

Une première observation est que la structure de la solution cherchée est donnée par les formules de Cramer.

Lemme 11.1 Le système possède une solution dont les coordonnées sont des fractions rationnelles. Ses numérateurs et dénominateurs ont degrés bornés par $nd-1$ et nd.

Démonstration. Le système (11.1) se récrit $A_1 y_1 + \cdots + A_n y_n = B$, où y_i est la $i$ème coordonnée de Y et A_i la $i$ème colonne de A. La formule de Cramer

$$\det(A_1, \ldots, A_{i-1}, B, A_{i+1}, \ldots, A_n) = y_i \det(A),$$

est obtenue en remplaçant B par sa valeur dans le membre gauche et en développant le déterminant.

Donc y_i est le quotient de déterminants de matrices appartenant à $\mathcal{M}_n(\mathbb{K}[X]_d)$, qui ont nécessairement degré au plus $nd-1$ pour le numérateur et nd pour le dénominateur. ■

L'Algorithme 11.1 de résolution est justifié par la complexité quasi-optimale des approximants de Padé (Chapitre 7). Dans tous les algorithmes qui vont suivre, c'est la recherche de série solution qui va dominer la complexité.

11.2 Développement comme une fraction rationnelle

La Proposition 3.5 (page 64) montre que la méthode de Newton fonctionne pour toute matrice inversible de séries. Avec cet algorithme, le calcul de la fraction rationnelle solution de (11.1) demande $O(n^\theta M(nd))$ opérations dans $\mathbb{K}$.

Il est possible d'améliorer l'efficacité lorsque la matrice est une matrice de polynômes. La base de cette amélioration est contenue dans le lemme suivant.

Lemme 11.2 Soit $A(X) \in \mathcal{M}_n(\mathbb{K}[X]_d)$ et $B(X) \in \mathcal{M}_{n,m}(\mathbb{K}[X]_{d-1})$, avec A inversible. Pour tout k, il existe une matrice $B_k \in \mathcal{M}_{n,m}(\mathbb{K}[X]_{d-1})$ telle que

$$A^{-1}B = a_0 + a_1 X + \cdots + a_{k-1}X^{k-1} + X^k A^{-1} B_k, \tag{11.2}$$

où $a_i \in \mathcal{M}_{n,m}(\mathbb{K})$.

Entrée A, B, k et $S = A^{-1} \bmod X^k$.
Sortie $a_0, \ldots, a_{k-1}$ et B_k définis par l'équation (11.2).
1. Calculer $SB =: a_0 + \cdots + a_{k-1}X^{k-1} \bmod X^k$.
2. Calculer $B_k = (B - A(a_0 + \cdots + a_{k-1}X^{k-1}))X^{-k}$.
3. Renvoyer $a_0, \ldots, a_{k-1}, B_k$.

Algorithme 11.2 – Développement de $A^{-1}B$.

Démonstration. Si les a_i sont les coefficients du développement en série de $A^{-1}B$, alors

$$B - A(a_0 + \cdots + a_{k-1}X^{k-1})$$

est une matrice de $\mathcal{M}_{n,m}(\mathbb{K}[X]_{d+k-1})$ qui, par construction, est divisible par X^k, d'où le lemme. ■

Ce résultat se traduit par l'Algorithme 11.2. La proposition suivante estime le coût de cet algorithme. Elle peut être vue comme l'analogue matriciel du Théorème 4.9 (page 88).

Proposition 11.3 Soit $A \in \mathcal{M}_n(\mathbb{K}[X]_d)$ avec $A(0)$ inversible, alors on peut calculer les $N \geq d$ premiers coefficients de A^{-1} en $O(n^\theta N\,M(d)/d)$ opérations dans $\mathbb{K}$.

Pour $B \in \mathcal{M}_{n,1}(\mathbb{K}[X]_{d-1})$, on peut calculer les $N \geq d$ premiers coefficients de $A^{-1}B$ en $O(n^2 N\,M(d)/d)$ opérations dans $\mathbb{K}$.

Démonstration. L'algorithme calcule d'abord l'inverse $S = A^{-1} \bmod X^d$ par l'algorithme de Newton, en $O(\mathsf{MM}(n,d)) = O(n^\theta M(d))$ opérations dans $\mathbb{K}$.

Ensuite, on applique N/d fois l'algorithme ci-dessus avec $k = d$ pour calculer à chaque itération d coefficients et un nouveau $B_{(i+1)d}$. Si on part de $B_0 = I$, le résultat fournit les N premiers coefficients de A^{-1} ; si $B_0 = B$, alors on obtient les coefficients de $A^{-1}B$.

Les deux étapes de l'algorithme (avec $k = d$) coûtent $\mathsf{MM}(n,d) = O(n^\theta M(d))$ opérations dans $\mathbb{K}$ si B a n colonnes, $O(n^2 M(d))$ s'il n'en a qu'une. ■

Avec cet algorithme, le calcul de la fraction rationnelle solution de (11.1) demande $O(n^3 M(d))$ opérations dans $\mathbb{K}$.

11.3 L'algorithme de Storjohann

L'algorithme de Storjohann permet de calculer les N premiers coefficients du développement en série de $A^{-1}B$ en $O(n^{\theta-1} N \log N\, M(d))$ opérations dans $\mathbb{K}$. Si $\theta < 3$, cette quantité croît avec n moins vite que la taille de l'inverse A^{-1}, qui n'est donc pas calculée en entier. Ainsi, la résolution du système linéaire (11.1) a un coût en $O(n^\theta M(d) \log(nd))$ opérations dans $\mathbb{K}$.

Entrée A, B, k et $S = [A^{-1}]_{k-2d+1}^{2d-2}$.
Sortie B_k défini par l'équation (11.2).
1. Calculer $U := [SB]_{k-d}^{d-1}$.
2. Calculer $B_k := [B - AU]_k^{d-1} X^{-k}$.
3. Renvoyer B_k.

Algorithme 11.3 – Développement de $A^{-1}B$ tronqué.

L'algorithme repose sur le développement de $A^{-1}B$ de la section précédente, joint d'une part à une technique de type « diviser pour régner », d'autre part au regroupement des calculs intermédiaires pour remplacer des groupes de n produits matrice-vecteur par des produits matrice-matrice.

Pour commencer, on peut modifier l'algorithme de développement de $A^{-1}B$ de la section précédente pour calculer B_k sans calculer tous les coefficients $a_0, \dots, a_{k-1}$. L'entrée nécessaire est moins grosse, et la complexité plus faible lorsque k est grand devant d. Pour une série $V = v_0 + v_1 X + \cdots$, on note

$$[V]_a^b := v_a X^a + \cdots + v_{a+b} X^{a+b},$$

avec la convention que les coefficients d'indice négatif de V sont nuls. Cela donne l'Algorithme 11.3.

Démonstration. Pour prouver la correction de cet algorithme, il faut s'assurer que les troncatures de séries sont suffisantes pour calculer le même polynôme B_k que précédemment.

Pour la première étape de l'algorithme, il suffit d'observer que B ayant degré au plus $d-1$, le coefficient de X^i dans $A^{-1}B$ ne dépend que de $[A^{-1}]_{i-d-1}^{d+1}$, pour tout i. Donc les coefficients calculés par cette première étape sont les mêmes que les a_i que précédemment, pour $i = k-d, \dots, k-1$.

Ensuite, il faut calculer $[X^k B_k]_k^{d-1}$. L'extraction des coefficients de l'équation (11.2) donne

$$[X^k B_k]_k^{d-1} = [B - A(a_0 + \cdots + a_{k-1}X^{k-1})]_k^{d-1} = [B - A(a_{k-d}X^{k-d} + \cdots + a_{k-1}X^{k-1})]_k^{d-1},$$

ce qui conclut la preuve. ■

Une observation importante concernant cet algorithme est que c'est le même polynôme S qui peut servir pour calculer B_{i+k} pour tout i. L'idée est alors de grouper plusieurs vecteurs B_i et de calculer les B_{i+k} correspondants par des produits matrice-matrice. Noter que la ressemblance entre cette idée et celle de l'algorithme de Keller-Gehrig en page 176.

Ainsi, si l'on connaît $B_0, B_{2m}, \dots, B_{2sm}$ et $[A^{-1}]_{m-2d}^{2d}$, alors le calcul des vecteurs suivants $B_m, B_{3m}, \dots, B_{(2s+1)m}$ ne requiert que $O(n^{\theta-1} s\, M(d))$ opérations dans $\mathbb{K}$.

En itérant cette idée, en partant de B_0 et en supposant connus $[A^{-1}]_{2^k-2d+1}^{2d-2}$ pour $k = 0, \dots, \lceil \log(N/d) \rceil =: k_{\max}$, on obtient d'abord $B_0, B_{2^{k_{\max}}}$, puis à l'étape suivante B_0, $B_{2^{k_{\max}-1}}$, $B_{2^{k_{\max}}}$, $B_{3 \cdot 2^{k_{\max}-1}}$, et ainsi de suite jusqu'à calculer toute la suite $B_0, B_d, B_{2d}, \dots$,

Entrée $S = [A^{-1}]_0^{d-1}$, $T = [A^{-1}]_{2^k-2d+1}^{2d-2}$ et B_{2^k-d} défini par (11.2) avec $B = I$.
Sortie $B_{2^{k+1}-d}$ et $[A^{-1}]_{2^{k+1}-2d+1}^{2d-2}$.
1. Calculer $[A^{-1}]_{2^{k+1}-2d+1}^{d-2} = [TB_{2^k-d}]_{2^{k+1}-2d+1}^{d-2}$.
2. Calculer $B_{2^{k+1}-d} := [-ATB_{2^k-d}]_{2^k}^{d-1}/X^{2^k}$.
3. Calculer $[A^{-1}]_{2^{k+1}-d}^{d-1} = [X^{2^{k+1}-d}B_{2^{k+1}-d}S]_{2^{k+1}-d}^{d-1}$.
4. Renvoyer $B_{2^{k+1}-d}$ et $[A^{-1}]_{2^{k+1}-2d+1}^{2d-2}$.

Algorithme 11.4 – Développement de A^{-1} — indices puissances de 2.

$B_{d\lceil N/d\rceil}$ en $O(k_{\max} n^{\theta-1} N\,M(d)/d)$ opérations dans $\mathbb{K}$. En multipliant alors ces vecteurs par $[A^{-1}]_0^d$ on obtient finalement les N premiers coefficients de $A^{-1}B$ pour le même coût.

Il reste à voir comment calculer les $[A^{-1}]_{2^k-2d-1}^{2d-2}$. Là encore, le point de départ est l'identité (11.2), avec $B = I$, $k = m$ et $k = p$:

$$\begin{aligned} A^{-1} &= a_0 + \cdots + a_{m-1}X^{m-1} + X^m A^{-1} B_m \\ &= a_0 + \cdots + a_{m-1}X^{m-1} + X^m(a_0 + \cdots + a_{p-1}X^{p-1} + X^p A^{-1} B_p)B_m. \end{aligned}$$

La seconde ligne est obtenue par substitution de la valeur de A^{-1} donnée par la première ligne avec $m = p$. Ces équations entraînent pour tout $\ell \geq 0$ tel que $m+p-\ell \geq d$

$$\begin{cases} B_{m+p} &= [-A[A^{-1}]_{p-2d+1}^{2d-2} B_m]_p^{d-1} X^{-p}, \\ [A^{-1}]_{m+p-\ell}^{\ell-1} &= [[A^{-1}]_{p-\ell-d+1}^{\ell+d-2} B_m]_{m+p-\ell}^{\ell-1}. \end{cases}$$

L'Algorithme 11.4 s'en déduit en utilisant cette identité avec $m = 2^k - d$, $p = 2^k$ et $\ell = d - 1$. Pour résumer, ces algorithmes mènent au résultat suivant.

Théorème 11.4 — Storjohann, 2002. Soient A une matrice $n \times n$ polynomiale de degré d avec $A(0)$ inversible et B un vecteur polynomial de degré au plus $d-1$, alors on peut calculer le numérateur et le dénominateur de $A^{-1}B$ en $O(n^\theta M(d)\log(nd))$ opérations dans $\mathbb{K}$.

L'énoncé suivant est un analogue du théorème 11.4 dans le cas des matrices à coefficients entiers. Sa preuve est toutefois bien plus technique et sera omise.

Théorème 11.5 — Storjohann, 2005. Soit A une matrice $n \times n$ inversible contenant des entiers de taille binaire au plus b et soit B un vecteur $n \times 1$ contenant des entiers de taille $O(b)$. On peut calculer le numérateur et le dénominateur de $A^{-1}B$ en $O(n^\theta \log(n) M(d)\log(d))$ opérations binaires, où $d = b + \log n$.

Notes

La conception d'algorithmes pour résoudre des systèmes linéaires à coefficients polynomiaux (resp. entiers) par développement de Taylor (resp. p-adique) et reconstruction rationnelle remonte aux travaux de Moenck et Carter [MC79] (dont sont issus l'Algorithme 11.1 et la Proposition 11.3) d'une part, et de Dixon [Dix82] d'autre part.

Dans tout ce texte, nous avons supposé que A(0) est inversible. En fait, les résultats s'étendent au cas où A est inversible sans que A(0) le soit. Il suffit de tirer aléatoirement un point et d'effectuer les développements en série au voisinage de ce point. L'algorithme devient probabiliste. Si la caractéristique n'est pas nulle, il se peut que A soit inversible sans que sa valeur en aucun point du corps ne le soit. On peut alors construire une extension du corps assez grande pour trouver un point où effectuer ces calculs. Ces considérations ont été prises en compte dans des travaux de Storjohann [Sto02 ; Sto03].

En 2003, Storjohann [Sto03] a montré que le déterminant d'une matrice polynomiale de taille n et degré au plus d pouvait se calculer en $O(n^\theta \mathsf{M}(d)\log^2 n)$ opérations arithmétiques. Storjohann et Villard [SV05] sont ensuite parvenus à calculer le rang et une base du noyau d'une telle matrice en $\tilde{O}(n^\theta d)$ opérations arithmétiques.

Pour n une puissance de 2, Jeannerod et Villard [JV05] ont donné un algorithme déterministe qui calcule l'inverse d'une matrice polynomiale inversible de taille n et degré au plus d génériquement en $\tilde{O}(n^3 d)$ opérations arithmétiques. Ils ont également étudié les réductions entre diverses opérations sur les matrices polynomiales en collaboration avec Giorgi [GJV03]. Plus récemment, Zhou, Labahn et Storjohann [ZLS15] ont amélioré le résultat de Jeannerod et Villard [JV05], en levant l'hypothèse de généricité et celle sur la taille n.

Storjohann a étendu tous les algorithmes de ce chapitre et tous ses autres résultats de 2002 et 2003 [Sto02 ; Sto03] au cas entier dans son article de 2005 [Sto05]. Il est le premier à être parvenu à montrer que le déterminant d'une matrice $n \times n$ formée d'entiers de taille binaire au plus b peut se calculer, à des facteurs logarithmiques près, aussi vite que le coût de la multiplication de deux telles matrices, plus exactement en $O(n^\theta \log(n)^2 \mathsf{M}(d)\log(d))$ opérations binaires, où $d = b+\log n$. En particulier, l'exposant du calcul du déterminant d'une matrice entière est bien égal à l'exposant ω du produit matriciel. Le meilleur algorithme précédemment connu, dû à Kaltofen et Villard [KV04] avait comme exposant l'expression exotique $\omega+\frac{1-\zeta}{\omega^2-(2+\zeta)\omega+2}$, où $\zeta \in [0,1]$ est un réel tel que la multiplication de deux matrices de taille $n \times n$ et $n \times n^\zeta$ puisse s'effectuer en $O(n^{2+o(1)})$ opérations.

Les techniques et les résultats de Storjohann ont été étendus récemment à d'autres opérations sur les matrices polynomiales et entières, avec un accent sur le caractère déterministe des algorithmes. Ainsi, Gupta, Sarkar, Storjohann et Valeriote [Gup+12] ont donné des versions déterministes du théorème 11.4, et du calcul rapide de certaines formes échelon. Pauderis et Storjohann [PS12] ont proposé un algorithme déterministe efficace pour déterminer si une matrice entière est unimodulaire.

Néanmoins, il reste encore des problèmes ouverts. Par exemple, il n'est toujours pas connu si l'on peut calculer le polynôme minimal ou le polynôme caractéristique d'une matrice polynomiale de taille n et degré au plus d en $\tilde{O}(n^\theta d)$ opérations arithmétiques, que ce soit en version probabiliste ou déterministe.

Bibliographie

Dix82 Dixon, John D. (1982). « Exact solution of linear equations using p-adic expansions ». In : *Numerische Mathematik*, vol. 40, n°1, p. 137–141.

GJV03 Giorgi, Pascal, Claude-Pierre Jeannerod et Gilles Villard (2003). « On the complexity of polynomial matrix computations ». In : *ISSAC'03 : International Symposium on Symbolic and Algebraic Computation*. Éd. par J. R. Sendra. New York : ACM Press, p. 135–142.

Gup+12 Gupta, Somit, Soumojit Sarkar, Arne Storjohann et Johnny Valeriote (2012). « Triangular x-basis decompositions and derandomization of linear algebra algorithms over $K[x]$ ». In : *Journal of Symbolic Computation*, vol. 47, n°4, p. 422–453.

JV05 Jeannerod, Claude-Pierre et Gilles Villard (2005). « Essentially optimal computation of the inverse of generic polynomial matrices ». In : *Journal of Complexity*, vol. 21, n°1, p. 72–86.

KV04 Kaltofen, Erich et Gilles Villard (2004). « On the complexity of computing determinants ». In : *Computational Complexity*, vol. 13, n°3-4, p. 91–130.

MC79 Moenck, Robert T. et John H. Carter (1979). « Approximate algorithms to derive exact solutions to systems of linear equations ». In : *Symbolic and Algebraic Computation. Eurosam '79, An International Symposium on Symbolic and Algebraic Manipulation, Marseille, France, June 1979*. Vol. 72. Lecture Notes in Computer Science. London, UK : Springer-Verlag, p. 65–73.

PS12 Pauderis, Colton et Arne Storjohann (2012). « Deterministic unimodularity certification ». In : *ISSAC 2012—Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation*. ACM, New York, p. 281–288.

Sto02 Storjohann, Arne (2002). « High-order lifting ». In : *ISSAC'02 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Teo Mora. ACM Press, p. 246–254.

Sto03 — (2003). « High-order lifting and integrality certification ». In : *Journal of Symbolic Computation*, vol. 36, n°3-4, p. 613–648.

Sto05 — (2005). « The shifted number system for fast linear algebra on integer matrices ». In : *Journal of Complexity*, vol. 21, n°4, p. 609–650.

SV05 Storjohann, Arne et Gilles Villard (2005). « Computing the rank and a small nullspace basis of a polynomial matrix ». In : *ISSAC'05 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 309–316.

ZLS15 Zhou, Wei, George Labahn et Arne Storjohann (2015). « A deterministic algorithm for inverting a polynomial matrix ». In : *Journal of Complexity*, vol. 31, n°2, p. 162–173.

12. Principe de transposition de Tellegen

Résumé

Le principe de transposition est un ensemble de règles de transformation pour les algorithmes calculant des applications linéaires. Pour tout algorithme qui calcule des produits matrice-vecteur par une matrice fixée $\mathbf{M}$, ces règles de transformation permettent d'obtenir un algorithme dual qui calcule des produits matrice-vecteur par la matrice transposée de $\mathbf{M}$. En outre, la complexité arithmétique de l'algorithme dual est essentiellement égale à celle de l'algorithme initial.

12.1 Introduction

Le *théorème de transposition de Tellegen* affirme que, étant donnée une matrice $\mathbf{M}$ de taille $m \times n$ à coefficients dans un corps $\mathbb{K}$, sans lignes ni colonnes nulles, tout algorithme linéaire $\mathcal{A}$ qui calcule l'application linéaire $\mathbf{v} \mapsto \mathbf{M} \cdot \mathbf{v}$ de $\mathbb{K}^n \to \mathbb{K}^m$ en L opérations dans $\mathbb{K}$ peut être transformé en un *algorithme dual* ${}^t\mathcal{A}$ qui calcule l'application linéaire transposée $\mathbf{w} \mapsto {}^t\mathbf{M} \cdot \mathbf{w}$ de $\mathbb{K}^m \to \mathbb{K}^n$ en $\mathrm{L} - n + m$ opérations dans $\mathbb{K}$. Ici, par *algorithme linéaire* on entend un algorithme qui n'utilise que des opérations linéaires en les éléments de l'entrée.

Par extension, on appelle *principe de Tellegen* l'ensemble des règles de transformation réalisant le passage de l'algorithme direct à l'algorithme dual.

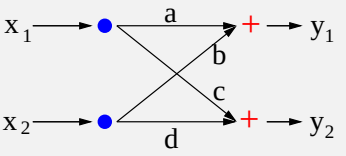
Motivation

Si l'algorithme utilisé pour la multiplication matrice-vecteur est l'algorithme naïf, cet énoncé devient trivial. En effet, dans ce cas :

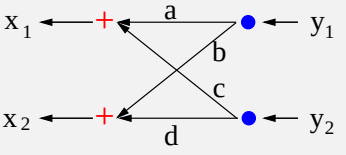
— $\mathbf{M} \cdot \mathbf{v}$ requiert $\sum_i (\alpha_i - 1) = \mathrm{E} - m$ opérations $\pm$ et E opérations $\times$,
— ${}^t\mathbf{M} \cdot \mathbf{w}$ requiert $\sum_j (\beta_j - 1) = \mathrm{E} - n$ opérations $\pm$ et E opérations $\times$,

où α_i (resp. β_j) est le nombre d'éléments non nuls de la i-ième ligne (resp. de la j-ième colonne) de $\mathbf{M}$, et E est le nombre d'éléments non nuls de $\mathbf{M}$. Par conséquent, pour une matrice complètement générique, le théorème de Tellegen ne présente aucun intérêt. Par contre, si la matrice $\mathbf{M}$ admet une structure, il est concevable que l'on dispose d'un algorithme plus rapide que l'algorithme naïf (quadratique) pour la multiplier par un vecteur de $\mathbb{K}^n$. En utilisant le principe de transposition, on obtient aussitôt un algorithme rapide pour multiplier ${}^t\mathbf{M}$ par un vecteur de $\mathbb{K}^m$.

Exemple 12.1 Considérons l'exemple suivant, qui illustre ce principe en utilisant la représentation par graphes des algorithmes linéaires. L'algorithme direct prend les valeurs x_1 et x_2 en entrée et renvoie $y_1 = ax_1 + bx_2$ et $y_2 = cx_1 + dx_2$ en sortie. Les arêtes représentent des multiplications par des valeurs scalaires a, b, c, d.



Transposer cet algorithme revient à inverser le flot du calcul, c'est-à-dire, inverser le sens des flèches, permuter les '+' avec les '•' et les entrées avec les sorties. L'algorithme ainsi obtenu prend y_1, y_2 en entrée et renvoie $x_1 = ay_1 + cy_2$ et $x_2 = by_1 + dy_2$. Il calcule donc bien l'application transposée de l'application initiale. De plus, le nombre d'opérations arithmétiques utilisées par les deux algorithmes est le même : 4 multiplications et 2 additions.



Utilité

Comme nous l'avons vu au Chapitre 10, beaucoup de problèmes en calcul formel s'expriment en termes d'algèbre linéaire structurée (multiplication par un vecteur ou résolution de système). Par exemple, les opérations élémentaires sur les polynômes (multiplication, division, évaluation-interpolation, interpolation rationnelle, extrapolation, etc.) sont *linéaires* si l'on fixe l'un des opérandes : ils se codent en termes de produits matrice-vecteur $\mathbf{M} \cdot \mathbf{v}$ ou $\mathbf{M}^{-1} \cdot \mathbf{v}$, où $\mathbf{M}$ est une matrice structurée (de type Toeplitz, Hankel, compagnon, Vandermonde, Cauchy, ...)

Grâce au principe de Tellegen, comprendre, analyser et améliorer un algorithme se ramène à comprendre, analyser et améliorer son transposé. Deux sont ses utilités principales : *trouver* des solutions algorithmiques de meilleure complexité, et *clarifier* le statut de certains algorithmes existant dans la littérature, qui parfois sont simplement des transposés d'algorithmes bien connus. Cela permet ainsi le traitement algorithmique unifié des problèmes duaux. On peut résumer en disant que le principe de transposition permet de diviser par deux le nombre d'algorithmes linéaires qu'il reste à découvrir.

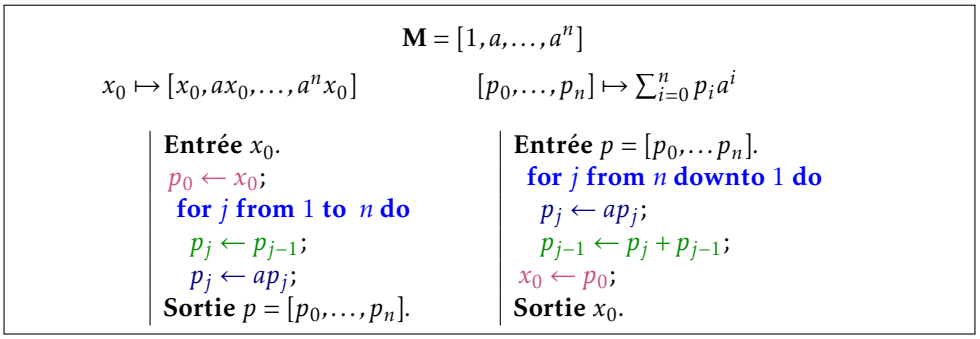


FIGURE 12.1 – Schéma de Horner et transposition de Tellegen : le schéma de Horner (à droite) est obtenu en transposant l'algorithme qui résout le problème dual (à gauche).

Un exemple moins trivial

Considérons le problème de l'évaluation d'un polynôme $P \in \mathbb{K}[X]$ de degré n en une valeur $a \in \mathbb{K}$. C'est une opération linéaire en les coefficients de P, de matrice $\mathbf{M} = [1, a, \ldots, a^n]$ dans les bases canoniques. Le problème transposé est donc le suivant : pour une valeur donnée $x_0 \in \mathbb{K}$, calculer les produits $a^i x_0$, pour $0 \leq i \leq n$. Pour ce problème, un algorithme naturel consiste à multiplier x_0 par a, ensuite multiplier le résultat par a, et ainsi de suite.

Le transposé de cet algorithme s'obtient en parcourant l'algorithme direct en sens inverse, tout en permutant les entrées avec les sorties, et en remplaçant chaque instruction par sa *transposée*, obtenue en appliquant un nombre restreint de règles syntaxiques. Dans ce processus, les boucles `for` montantes deviennent des boucles `for` descendantes.

De cette manière, on obtient *automatiquement* un algorithme pour le problème de départ, à savoir, l'évaluation de P sur a. Dans notre cas, il s'avère que l'algorithme transposé coïncide avec la fameuse *méthode de Horner*, voir la Figure 12.1. Observons que l'algorithme transposé utilise n opérations de plus que l'algorithme direct. Cette perte s'explique par le théorème de Tellegen : il s'agit tout simplement de la différence entre le nombre de colonnes et le nombre de lignes de $\mathbf{M}$. Cette observation peut être utilisée pour expliquer l'optimalité de la règle de Horner.

12.2 La version en termes de graphes du principe de Tellegen

Dans cette section nous donnons une version du théorème de Tellegen dans un modèle particulier, celui des graphes de calcul (DAG).

Définition 12.1 Un *graphe orienté acyclique*, ou DAG (de l'anglais *directed acyclic graph*) est un graphe orienté $G = (V, E)$ qui ne possède pas de cycle. Un DAG $\mathbb{K}$-linéaire est un DAG muni d'une fonction de poids $\lambda : E \to \mathbb{K}$.

Soit $I = \{x_1, \ldots, x_n\}$ l'ensemble des nœuds d'entrée de G. À tout sommet $v \in V$ on

associe une forme linéaire dans $\mathbb{K}[X_1,\dots,X_n]$ de la façon suivante :
— si $v = x_i \in I$, alors $h_v := X_i$,
— si $v \in V \setminus I$, alors $h_v := \sum_{e=(w,v)\in E} \lambda(e)\, h_w$.

Définition 12.2 On dit que G calcule la matrice $\mathbf{M} = [a_{i,j}]$ de taille $m \times n$ si les formes linéaires associées aux nœuds de sortie sont $F_i = \sum_{j=1}^{n} a_{i,j} X_j$, $i = 1,\dots,m$. Le coût de G est le nombre d'opérations linéaires $c(G)$ induites par G.

Avec ces notations, le principe de Tellegen s'énonce comme suit.

Théorème 12.1 Soit G un $\mathbb{K}$-DAG qui calcule une matrice $\mathbf{M}$ de $\mathcal{M}_{m,n}(\mathbb{K})$. Alors le graphe transposé ${}^t G$, obtenu en inversant le sens des flèches sans changer leur poids, calcule la matrice ${}^t\mathbf{M}$. De plus,

$$c({}^t G) = c(G) - n + m.$$

Démonstration. La forme linéaire calculée par un sommet v de G est

$$h_v = \sum_{j=1}^{n} \left(\sum_{p \in \mathrm{Chemin}(x_j, v)} \lambda(p) \right) X_j, \quad \text{où} \quad \lambda(p) = \prod_{e \text{ arête de } p} \lambda(e).$$

On a donc l'égalité $a_{i,j} = \sum_{p \in \mathrm{Chemin}(x_j, F_i)} \lambda(p)$, qui montre que ${}^t G$ calcule bien ${}^t\mathbf{M}$.

L'égalité entre les coûts se déduit du fait que la quantité $c(G) - \mathrm{card}(I(G))$ ne dépend pas de l'orientation de G, comme montré par le lemme suivant. ■

Lemme 12.2 — Formule pour le coût d'un graphe de calcul. Avec les notations précédentes, on a l'égalité

$$c(G) = \mathrm{card}\Big(\{e \in E \mid \lambda(e) \neq \pm 1\}\Big) + \mathrm{card}(E) - \mathrm{card}(V) + \mathrm{card}(I).$$

Démonstration. Tout sommet v qui reçoit $d(v) > 0$ arêtes contribue au coût de G par $\mathrm{card}\Big(\{e = (w,v) \in E \mid \lambda(e) \neq \pm 1\}\Big) + \Big(d(v) - 1\Big)$ opérations dans $\mathbb{K}$.
Graphiquement, cela se voit sur la représentation de la Figure 12.2. La conclusion se déduit alors aisément de la suite d'égalités

$$\sum_{v \in V \setminus I} \Big(d(v) - 1\Big) = \sum_{v \in V \setminus I} d(v) - \sum_{v \in V \setminus I} 1 = \mathrm{card}(E) - (\mathrm{card}(V) - \mathrm{card}(I)).$$ ■

12.3 Principe de Tellegen pour les programmes linéaires

Le but de cette section est de décrire le principe de Tellegen dans le modèle des programmes linéaires sur des machines à allocation de registres (machines à accès direct, MAD, ou random access memory, RAM, en anglais).

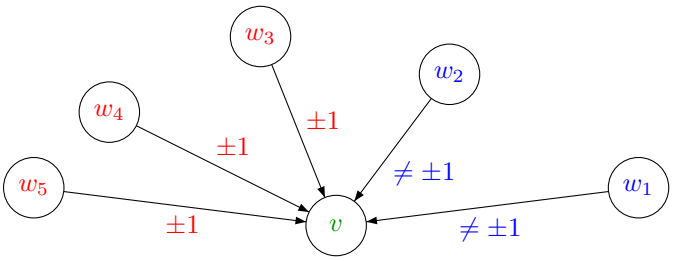


FIGURE 12.2 – Contribution d'une arête au coût d'un graphe de calcul.

Machines à registres

Commençons par définir les machines MAD à instructions linéaires.

Définition 12.3 On fixe un entier M ; il représente la mémoire disponible, de sorte que l'on peut accéder à M registres $R_1, \ldots, R_M$. On y stocke des nombres, c'est-à-dire des éléments du corps $\mathbb{K}$.

Un *programme linéaire* est la donnée des objets suivants :

— un sous-ensemble $R_{i_1}, \ldots, R_{i_n}$ des registres que l'on appelle *entrée* ;
— un sous-ensemble $R_{o_1}, \ldots, R_{o_m}$ des registres que l'on appelle *sortie* ;
— une suite d'instructions portant sur les registres, choisies parmi :
  — $R_i = \pm R_j \pm R_k$, avec $1 \leq i, j, k \leq M$,
  — $R_i = \lambda R_j$, avec $1 \leq i, j \leq M$ et λ dans $\mathbb{K}$.

Le fonctionnement d'un tel programme est le suivant : à l'initialisation, les registres d'entrée reçoivent des valeurs $x_1, \ldots, x_n \in \mathbb{K}$ et les autres sont mis à zéro. On effectue ensuite toutes les instructions, puis le programme renvoie les valeurs des registres de sortie. Noter que l'on calcule bel et bien une fonction linéaire des x_i.

Transposition de programme

Cas d'un jeu d'instructions réduit

Pour commencer, on traite le cas d'une machine avec un jeu d'instructions limité par rapport au cas général. On ne s'autorise que les instructions du type :

— $R_i = R_i \pm R_j$, avec $1 \leq i, j \leq M$;
— $R_i = \lambda R_i$, avec $1 \leq i \leq M, \lambda \in \mathbb{K}$.

Pour « transposer » un programme comportant des instructions de ce type, on interprète chacune de ces instructions comme une application linéaire $\mathbb{K}^M \to \mathbb{K}^M$. Pour motiver ce procédé, considérons un exemple. Avec $M = 3$, l'instruction $R_1 = R_1 + R_3$ s'interprète comme l'application linéaire dont la matrice est

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix};$$

R_2 et R_3 n'ont pas changé et R_1 devient $R_1 + R_3$. La transposée de cette application linéaire a pour matrice

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix};$$

on en déduit que l'on peut la traduire par l'instruction $R_3 = R_3 + R_1$.

De manière générale, la transposée de l'instruction $R_i = R_i + R_j$ est l'instruction $R_j = R_j + R_i$. Par écriture matricielle, on voit que l'instruction $R_i = \lambda R_i$ est inchangée par transposition, et que $R_i = R_i - R_j$ se transpose en $R_j = R_j - R_i$. On peut alors définir le programme transposé d'un programme.

Définition 12.4 Le *programme transposé* ${}^t\mathcal{P}$ du programme linéaire $\mathcal{P}$ est donné par :
- les entrées de ${}^t\mathcal{P}$ sont les sorties de $\mathcal{P}$;
- les sorties de ${}^t\mathcal{P}$ sont les entrées de $\mathcal{P}$;
- les instructions de ${}^t\mathcal{P}$ sont les transposées des instructions de $\mathcal{P}$, prises en sens inverse.

La dernière condition, le retournement de la liste des instructions, traduit l'égalité matricielle ${}^t(AB) = {}^tB \cdot {}^tA$. La définition montre que si $\mathcal{P}$ calcule une application linéaire $\mathbb{K}^n \to \mathbb{K}^m$, alors ${}^t\mathcal{P}$ calcule bien l'application transposée $\mathbb{K}^m \to \mathbb{K}^n$.

Exemple 12.2 Considérons le programme

$R_4 = R_4 + R_2$
$R_3 = 3 \times R_3$
$R_4 = R_4 + R_3$
$R_2 = 2 \times R_2$
$R_3 = R_3 + R_2$
$R_3 = R_3 + R_1$

dont les entrées sont R_1, R_2, R_3 et les sorties R_3, R_4. Ce programme calcule l'application linéaire dont la matrice est

$$\begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 3 \end{pmatrix}.$$

Le programme transposé s'écrit

$R_1 = R_1 + R_3$
$R_2 = R_2 + R_3$
$R_2 = 2 \times R_2$
$R_3 = R_3 + R_4$
$R_3 = 3 \times R_3$
$R_2 = R_2 + R_4$

et on vérifie qu'il calcule l'application linéaire dont la matrice est

$$\begin{pmatrix} 1 & 0 \\ 2 & 1 \\ 3 & 3 \end{pmatrix}.$$

Cas général

Le cas du jeu d'instructions général se ramène au cas du jeu d'instructions réduit de manière immédiate. Ainsi, l'instruction $R_1 = R_2 + R_3$ est équivalente à la suite d'instructions

$$\begin{aligned} R_1 &= 0 \\ R_1 &= R_1 + R_2 \\ R_1 &= R_1 + R_3 \end{aligned}$$

Il est donc possible de le transposer sous la forme

$$\begin{aligned} R_3 &= R_3 + R_1 \\ R_2 &= R_2 + R_1 \\ R_1 &= 0 \end{aligned}$$

De même, on récrit aisément l'instruction $R_i = \lambda R_j$ en utilisant le jeu d'instructions réduit, ce qui permet de la transposer, ...

Optimisations

Notre principe de transposition n'est pas complètement satisfaisant : au vu des règles de récriture ci-dessus, il semble que l'on puisse perdre jusqu'à un facteur 3 (en termes de nombre de lignes). On peut optimiser le code produit, afin de regagner, autant que possible, les lignes perdues. On utilise pour ce faire les règles suivantes :

Suppression des zéros La transformation pour passer du cas général au jeu d'instructions réduit introduit des lignes du type $R_i = 0$. On peut supprimer une telle ligne, à condition de récrire les instructions suivantes en conséquence.

Suppression des recopies Après avoir supprimé les zéros, il se peut qu'il reste des lignes du type $R_i = \pm R_j$. On peut également supprimer ce type de ligne, à condition de récrire de manière judicieuse les instructions qui suivent.

Muni de ces règles supplémentaires, on peut montrer qu'il est possible d'obtenir un code transposé qui fait exactement le même nombre de lignes que l'original mais cela dépasse le cadre de cet ouvrage.

12.4 Applications

Dans cette section, nous illustrons l'utilité pratique du principe de Tellegen à travers quelques exemples. Les techniques de transposition permettront d'engendrer des programmes linéaires pour effectuer les opérations suivantes sur les polynômes à une variable : multiplication, division euclidienne, évaluation et interpolation multipoint. La Figure 12.3 contient une liste (non exhaustive) d'opérations linéaires de base sur les polynômes à une variable et leurs problèmes transposés. Tout algorithme résolvant un problème de la colonne de gauche peut être transposé en un algorithme de même complexité pour le problème correspondant de la colonne de droite.

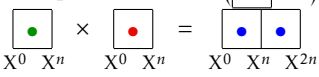
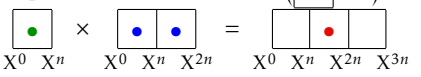
Problème direct	Problème transposé
multiplication $\mathrm{mul}(\square,)$ 	produit médian $\mathrm{mul}^{\mathrm{t}}(\square,)$ 
division euclidienne	extension de récurrences
$A \mapsto A \bmod P$	$(a_0,\dots,a_{n-1}) \mapsto (a_0,\dots,a_{2n-1})$
évaluation multipoint	sommes de Newton pondérées
$P \mapsto (P(a_0),\dots,P(a_{n-1}))$	$(p_0,\dots,p_{n-1}) \mapsto (\sum p_i,\dots,\sum p_i a_i^{n-1})$
interpolation	décomposition en éléments simples
(systèmes de Vandermonde)	(systèmes de Vandermonde transposés)
décalage de polynômes	évaluation de factorielles descendantes
$P(X) \mapsto P(X+1)$	$P = \sum a_i X^{\underline{i}} \mapsto (P(0),\dots,P(n-1))$
extrapolation sur $0,1,2,\dots$	division modulo $(X-1)^n$
q-décalage	évaluation de q-factorielles descendantes
composition modulaire	projection des puissances

FIGURE 12.3 – « Dictionnaire de Tellegen » pour les polynômes à une variable.

Produit médian des polynômes

Commençons par transposer la multiplication des polynômes. Cette opération devient linéaire lorsqu'on fixe l'un des deux opérandes. Fixons une fois pour toutes un polynôme f dans $\mathbb{K}[X]$ de degré m. Pour $n \geq 0$ quelconque, considérons l'application de multiplication $m_{f,n} : \mathbb{K}[X]_{\leq n} \to \mathbb{K}[X]_{\leq m+n}$ qui à un polynôme g associe le produit fg. Il est aisé de voir que la transposée de cette opération ${}^t m_{f,n} : \mathbb{K}[X]_{\leq m+n} \to \mathbb{K}[X]_{\leq n}$ consiste à extraire les coefficients de $X^m, X^{m+1}, \dots, X^{m+n}$ du produit d'un polynôme de degré au plus $m+n$ par le polynôme réciproque $\tilde{f}$ de f.

Exemple 12.3 Par exemple, soit $f = 2 + X + 3X^2$; la matrice de $m_{f,2}$ dans les bases canoniques de $\mathbb{K}[X]_{\leq 2}$ et $\mathbb{K}[X]_{\leq 4}$ est donnée par la matrice de type Toeplitz

$$\begin{pmatrix} 2 & 0 & 0 \\ 1 & 2 & 0 \\ 3 & 1 & 2 \\ 0 & 3 & 1 \\ 0 & 0 & 3 \end{pmatrix}.$$

Sa transposée est

$$\begin{pmatrix} 2 & 1 & 3 & 0 & 0 \\ 0 & 2 & 1 & 3 & 0 \\ 0 & 0 & 2 & 1 & 3 \end{pmatrix}.$$

Cette dernière matrice est la partie médiane de la matrice de Toeplitz suivante

$$\begin{pmatrix} 3 & 0 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 0 \\ \mathbf{2} & \mathbf{1} & \mathbf{3} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{2} & \mathbf{1} & \mathbf{3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{2} & \mathbf{1} & \mathbf{3} \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix},$$

qui représente le produit du polynôme $\tilde{f} = 3 + X + 2X^2$ par un polynôme h de degré 4. Cette partie médiane donne les coefficients de degré $2, 3, 4$ du produit $\tilde{f}h$. Cela explique pourquoi le produit transposé est appelé *produit médian*.

Le produit médian est une opération importante, qui intervient comme brique de base dans nombre d'algorithmes sur les polynômes et les séries. Son calcul efficace est un problème important à résoudre avant de passer à la transposition d'autres algorithmes plus sophistiqués.

Par exemple, l'utilisation du produit médian permet d'améliorer, d'un facteur constant, l'efficacité de l'opérateur de Newton pour l'inverse de séries formelles, et mène aux optimisations mentionnées en page 63. Cette remarque s'applique également aux autres opérations rapides sur les séries décrites au Chapitre 3. L'accélération se répercute aussi sur la division euclidienne rapide, qui repose sur une inversion de série (Théorème 4.1). En vertu du principe de Tellegen (Théorème 12.1) appliqué aux matrices de Toeplitz, le coût du produit médian est celui du produit fg, à m opérations près. En particulier, si $m = n$, cela permet d'effectuer le calcul de la partie médiane pour le coût d'une multiplication en degré n, au lieu des deux qu'on attendrait naïvement. Il y a plusieurs algorithmes pour multiplier des polynômes, à chacun correspond donc un algorithme de même complexité pour calculer le produit transposé ; cet algorithme transposé peut être obtenu en appliquant le principe de Tellegen, soit dans sa version graphes, soit par transformation de programmes linéaires. Plus exactement, on a le résultat général suivant.

Théorème 12.3 Tout algorithme de complexité arithmétique $\mathsf{M}(n)$ pour la multiplication polynomiale en degré n peut être transformé en un algorithme de complexité arithmétique $\mathsf{M}(n) + \mathrm{O}(n)$ pour le produit médian en degrés $(n, 2n)$.

Exercice 12.1 Vérifier le théorème dans le cas de la multiplication naïve des polynômes. Expliciter l'algorithme transposé dans ce cas. ■

Exemple 12.4 En Figure 12.4, à gauche, est représenté un DAG qui calcule le produit de deux polynômes de degré 1 *à la Karatsuba* (en utilisant 3 multiplications au lieu de 4). Par transposition, on déduit un algorithme *à la Karatsuba* pour le produit transposé de deux polynômes de degré 1 et 2. L'algorithme direct utilise 6

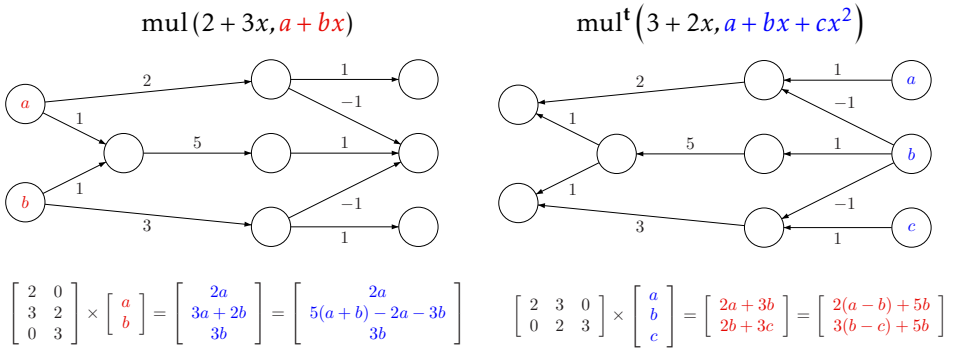


FIGURE 12.4 – Produit direct et transposé *à la Karatsuba*, version DAG.

opérations arithmétiques (3 additions et 3 multiplications) ; l'algorithme transposé en utilise 7 (4 additions et 3 multiplications). La différence de 1 opération est bien sûr prédite par le Théorème 12.1. L'algorithme de Karatsuba et son code transposé sont donnés en Figure 12.5. Tous les deux ont la même complexité $O(n^{\log 3})$.

Exemple 12.5 La matrice de la DFT étant symétrique, à tout algorithme (linéaire) qui la calcule est associé un algorithme transposé de même coût qui la calcule également. Par exemple, à la variante de la DFT décrite au Chapitre 2 (due à Gentleman et Sande) correspond un algorithme de Cooley et Tukey. En Figure 12.6 sont représentés les DAG de ces algorithmes pour une DFT sur 4 points.

Division avec reste et extension des récurrences à coefficients constants

Le dual du problème de la division avec reste d'un polynôme de degré N par un polynôme fixé de degré n est l'extension des récurrences linéaires à coefficients constants. Étant donnés les n premiers termes d'une suite qui vérifie une récurrence linéaire à coefficients constants d'ordre n, il s'agit de calculer les N termes suivants.

Pour une récurrence d'ordre $n = 1$, la preuve est immédiate : étendre une récurrence $u_{n+1} = au_n$, de condition initiale $u_0 = x_0$ revient à calculer la suite des valeurs $x_0, ax_0, \ldots, a^N x_0$ à partir de x_0. Or, on a vu en page 219 que le dual de ce problème est l'évaluation polynomiale en a, autrement dit, la division avec reste modulo $X - a$.

Exercice 12.2 Finir la preuve dans le cas général (n quelconque). ∎

Exercice 12.3 Transposer l'algorithme rapide de division euclidienne donné dans la preuve du Théorème 4.1 (page 83). En déduire un algorithme de complexité $O(M(n))$ permettant de calculer les $2n$ premiers termes d'une suite donnée par une récurrence linéaire d'ordre n à coefficients constants et n conditions initiales. ∎

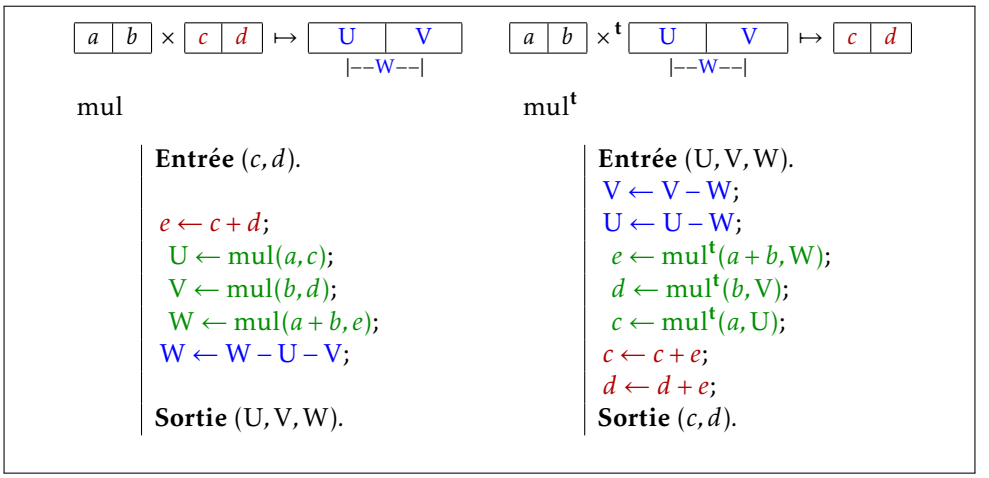


FIGURE 12.5 – Algorithme de Karatsuba et son transposé.

Évaluation multipoint et interpolation

Comme vu au Chapitre 5, l'évaluation multipoint se traduit en termes matriciels par un produit entre la matrice de Vandermonde associée aux points a_i et le vecteur des coefficients du polynôme P, voir la Figure 12.7. Ainsi, le problème transposé est la multiplication d'une matrice de Vandermonde transposée par un vecteur, c'est-à-dire, le calcul de sommes de Newton pondérées (on les appelle ainsi car si tous les p_i valent 1, on retrouve les sommes des puissances des a_i). Il est possible d'exploiter algorithmiquement cette dualité, à l'aide du principe de transposition. L'idée est de proposer d'abord un algorithme rapide pour le calcul de ces sommes de Newton, et ensuite de le transposer.

Le point de départ est la série génératrice des sommes de Newton pondérées

$$\sum_{s\geq 0}\Big(\sum_{i=0}^{n-1} p_i a_i^s\Big)\mathrm{X}^{-s-1}$$

dont on observe qu'elle est *rationelle* et vaut Q = B/A, où

$$\mathrm{A} = (\mathrm{X}-a_0)\cdots(\mathrm{X}-a_{n-1}) \quad \text{et} \quad \mathrm{B} = \sum_{i=0}^{n-1} p_i \frac{\mathrm{A}}{\mathrm{X}-a_i}.$$

D'où l'algorithme suivant : on calcule A et B et on retourne les n premiers coefficients du développement en série de Taylor à l'infini de Q = B/A. Le calcul de A se fait en utilisant l'Algorithme 5.3 (page 100), de complexité $\frac{1}{2}\mathsf{M}(n)\log n + \mathrm{O}(\mathsf{M}(n))$. Celui de B peut se faire par l'algorithme de type « diviser pour régner » représenté en en haut de la Figure 12.8, de complexité $\mathsf{M}(n)\log n + \mathrm{O}(\mathsf{M}(n))$. Une solution équivalente serait de calculer simultanément A et B par l'Algorithme 5.6 en page 103. Enfin, le

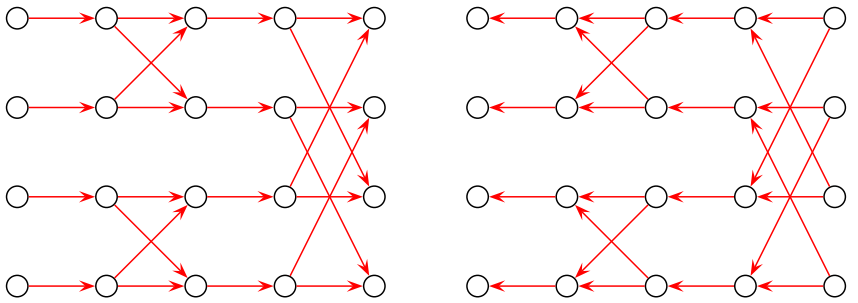


FIGURE 12.6 – Dualité entre deux classes d'algorithmes pour la DFT : à gauche, l'algorithme *decimation-in-time* de Cooley–Tukey ; à droite l'algorithme *decimation-in-frequency* de Gentleman–Sande.

$$\begin{pmatrix} 1 & a_0 & \cdots & a_0^n \\ 1 & a_1 & \cdots & a_1^n \\ \vdots & \vdots & & \vdots \\ 1 & a_n & \cdots & a_n^n \end{pmatrix} \cdot \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{pmatrix} = \begin{pmatrix} \mathrm{P}(a_0) \\ \mathrm{P}(a_1) \\ \vdots \\ \mathrm{P}(a_n) \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} 1 & 1 & \cdots & 1 \\ a_0 & a_1 & \cdots & a_n \\ \vdots & \vdots & & \vdots \\ a_0^n & a_1^n & \cdots & a_n^n \end{pmatrix} \cdot \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{pmatrix} = \begin{pmatrix} \sum p_i \\ \sum a_i p_i \\ \vdots \\ \sum a_i^n p_i \end{pmatrix}.$$

FIGURE 12.7 – Évaluation multipoint et sa transposée. Sommes de Newton pondérées.

développement en série de Q s'effectue en $\mathrm{O}(\mathsf{M}(n))$ opérations arithmétiques grâce à des itérations de Newton, comme décrit au Chapitre 3.

En résumé, on obtient un algorithme pour le calcul des sommes de Newton pondérées, de complexité

$$\frac{3}{2}\mathsf{M}(n)\log n + \mathrm{O}\big(\mathsf{M}(n)\big).$$

Par transposition, on construit un algorithme *de même complexité* pour l'évaluation multipoint. Cet algorithme est représenté graphiquement en bas de la Figure 12.8. Tout comme l'Algorithme 5.5 (page 101), le nouvel algorithme repose sur une stratégie « diviser pour régner », mais remplace, à chaque niveau de récursion, les divisions avec reste par des produits médians, moins coûteux, d'où le gain d'un facteur constant dans la complexité.

Exercice 12.4 Écrire les détails des deux algorithmes en Figure 12.8. ■

Exercice 12.5 Soit $\mathbb{K}$ un corps et soit n un entier. On considère le problème suivant : Étant données les valeurs en $0, 1, \ldots, n-1$ d'un polynôme (inconnu) de $\mathbb{K}[X]$ de degré au plus n, calculer les valeurs prises par ce polynôme en $n, n+1, \ldots, 2n-1$.

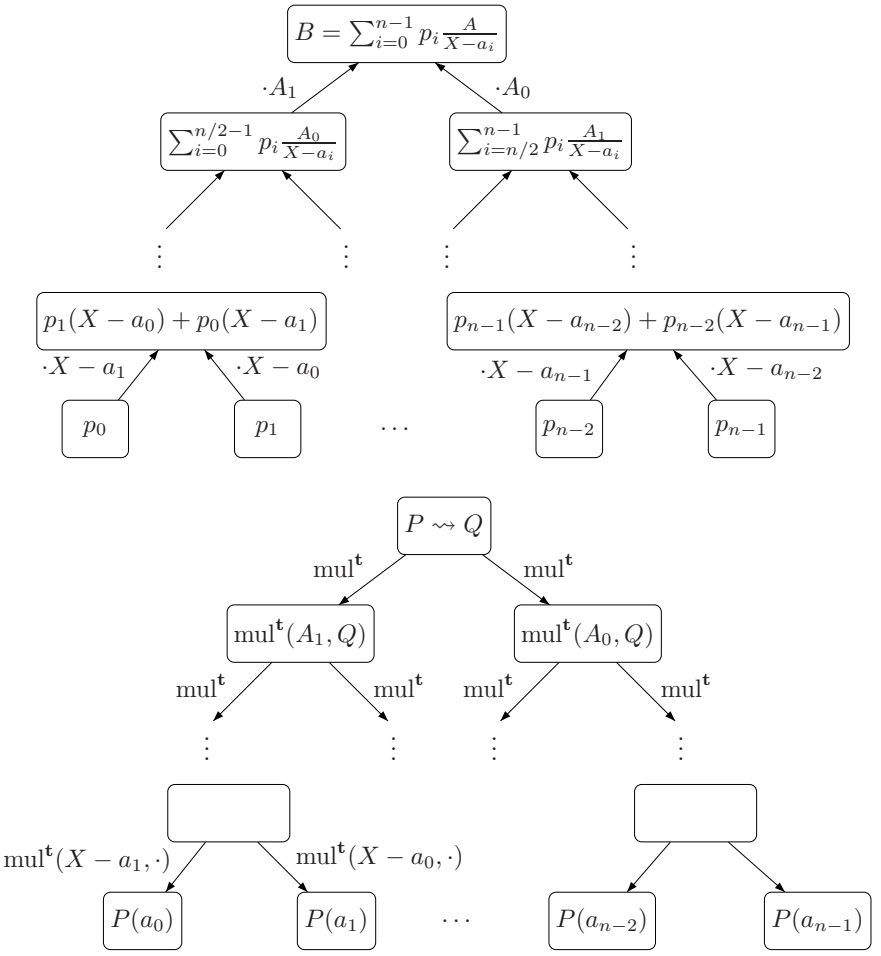


FIGURE 12.8 – Transposition du calcul des sommes de Newton pondérées en un calcul rapide d'évaluation multipoint : l'algorithme rapide d'évaluation multipoint par produits médians répétés (en bas) est obtenu par transposition d'un algorithme rapide pour les sommes de Newton pondérées (en haut).

Déterminer le problème dual, donner un algorithme de complexité $O(M(n))$ pour ce dernier, et en déduire un algorithme explicite de complexité $O(M(n))$ pour le problème de départ. ∎

L'évaluation et l'interpolation ont des coûts équivalents

Un dernier exemple d'application du théorème de Tellegen, de nature plus théorique, est la preuve du fait que les problèmes d'évaluation multipoint et d'interpolation polynomiale sont équivalents du point de vue de la complexité.

Théorème 12.4 Tout algorithme qui effectue l'évaluation multipoint (resp. l'interpolation) sur une famille fixée de points peut être transformé en un algorithme d'interpolation (resp. d'évaluation multipoint) sur la même famille de points, de même complexité à un nombre constant de multiplications polynomiales près.

Autrement dit, si $\mathsf{E}(n)$ et $\mathsf{I}(n)$ représentent les complexités de l'évaluation multipoint et de l'interpolation en n points, alors :

$$\mathsf{I}(n) \in \mathrm{O}\Big(\mathsf{E}(n) + \mathsf{M}(n)\Big) \quad \text{et} \quad \mathsf{E}(n) \in \mathrm{O}\Big(\mathsf{I}(n) + \mathsf{M}(n)\Big).$$

Démonstration. Supposons qu'on dispose d'un algorithme $\mathcal{I}$ de complexité $\mathsf{I}(n)$ pour l'interpolation sur les points $\mathbf{a} = (a_0, \ldots, a_{n-1})$. Le théorème de Tellegen montre qu'on peut construire un algorithme ${}^t\mathcal{I}$ de complexité $\mathsf{I}(n)$ pour l'*interpolation transposée* sur $\mathbf{a}$. On va montrer comment en déduire un algorithme $\mathcal{E}$ pour l'évaluation multipoint sur $\mathbf{a}$. L'idée est d'utiliser la factorisation matricielle

$$\mathbf{V_a} = ({}^{\mathbf{t}}\mathbf{V_a})^{-1} \cdot \mathbf{H_a}, \quad \text{où} \quad \mathbf{H_a} = \begin{pmatrix} n & \sum_i a_i & \cdots & \sum_i a_i^{n-1} \\ \sum_i a_i & \sum_i a_i^2 & \cdots & \sum_i a_i^n \\ \vdots & \vdots & & \vdots \\ \sum_i a_i^{n-1} & \sum_i a_i^{n-1} & \cdots & \sum_i a_i^{2(n-1)} \end{pmatrix},$$

qui suggère l'algorithme $\mathcal{E}$ suivant :

- appliquer $\mathcal{I}$ pour calculer $\mathrm{A}(\mathrm{X}) = \prod_i (\mathrm{X} - a_i)$, en complexité $\mathsf{I}(n)$;
- calculer $\mathbf{H_a}$ à partir des $2n-1$ premières sommes de Newton de $\mathrm{A}(\mathrm{X})$, par l'algorithme de complexité $\mathrm{O}(\mathsf{M}(n))$ donné en Proposition 3.10, page 67 ;
- calculer $\mathbf{v} = \mathbf{H_a} \cdot \mathbf{p}$ en $\mathrm{O}(\mathsf{M}(n))$ opérations, comme au Chapitre 10 ;
- renvoyer $\mathbf{V_a} \cdot \mathbf{p} = {}^t(\mathbf{V_a^{-1}}) \cdot \mathbf{v}$ en $\mathsf{I}(n)$ opérations en utilisant l'algorithme ${}^t\mathcal{I}$.

La preuve du sens inverse est laissée en exercice. ■

Exercice 12.6 Finir la preuve du Théorème 12.4. ■

Notes

La présentation de la Section 12.2 est inspirée d'un article de Kaminski, Kirkpatrick et Bshouty [KKB88]. L'algorithme rapide de l'Exercice 12.3 pour l'extension des récurrences est dû à Shoup [Sho91a ; Sho99]. La dualité *division polynomiale* ↔ *extension de récurrences* de la Section 12.4 est implicite dans l'article de Cerlienco, Mignotte, et Piras de 1987 [CMP87, §IV.3]. Elle permet de clarifier le statut de l'algorithme de Shoup : c'est la transposée de l'algorithme de Strassen [Str73] pour la division euclidienne des polynômes, fait mis en évidence par Bostan, Lecerf et Schost [BLS03].

L'algorithme présenté en Section 12.4, améliorant (d'un facteur constant) les algorithmes classiques d'évaluation, est dû à Bostan, Lecerf, et Schost [BLS03]. Le principe de transposition est également employé par ces auteurs pour accélérer (toujours par des facteurs constants) les algorithmes classiques pour l'interpolation et pour son problème dual (la résolution de systèmes de Vandermonde transposés). Les mêmes méthodes s'étendent au cas plus général du théorème des restes chinois [Bos+04a].

Le Théorème 12.4 est dû à Bostan et Schost [BS04]. La factorisation matricielle utilisée dans sa preuve est tirée de l'article de Canny, Kaltofen et Yagati de 1989 [CKY89].

Une autre application récente du principe de transposition est la suivante [BSS08] : si $(P_\ell)_{\ell\geq 0}$ est une famille de polynômes de $\mathbb{K}[X]$, avec $\deg(P_\ell) = \ell$, dont la série génératrice exponentielle $\sum_{\ell\geq 0} P_\ell(X)/\ell! \cdot Y^\ell$ est de la forme [1] $v(Y) \cdot \exp(X \cdot h(Y))$, alors les conversions entre la base canonique $(X^\ell)_{\ell\geq 0}$ de $\mathbb{K}[X]$ et la base $(P_\ell)_{\ell\geq 0}$ peuvent s'effectuer en $O(M(n))$ opérations dès lors que v mod Y^n peut se calculer en $O(M(n))$ opérations, et pour une large classe de séries h.

Historique

Le principe de transposition a une histoire longue et tortueuse. Cette technique de transformation des algorithmes linéaires est issue du domaine des circuits électroniques [Ant79 ; Bor56 ; PSD70] et de la théorie du contrôle [Kal59]. Le principe même remonte aux années 1950 ; on en trouve les traces dans un article de Tellegen [Tel52] sur les réseaux électriques. Il a été généralisé aux filtres digitaux par Fettweis [Fet71], où l'on trouve une version embryonnaire de la version par graphes. Le théorème de Tellegen a été redémontré plusieurs fois, dans divers contextes et degrés de généralité, par Fiduccia [Fid72a ; Fid73], Hopcroft et Musinski [HM73], Knuth et Papadimitriou [KP81], et par Kaminski, Kirkpatrick et Bshouty [KKB88]. En calcul formel, il a été popularisé dans les travaux de Ben-Or, Tiwari [BT88], Kaltofen, Canny, Lakshman [CKY89 ; KCJ00], Shoup [Sho91a ; Sho95 ; Sho99], Lecerf, Schost [LS03], Hanrot, Quercia, Zimmermann [HQZ04], Zippel [Zip90], von zur Gathen et Gerhard [GG99], … Il est resté longtemps méconnu, comme le montre cet extrait de l'article de Wiedemann [Wie86] de 1986 : « I was not able to find an example of a linear operator that is easy to apply but whose transpose is difficult to apply ». Le nom *principe de transposition* semble avoir été introduit par Kaltofen et Shoup au début des années 1990 [Kal93b ; Sho94]. Un point de vue légèrement différent sur les aspects historiques liés au principe de transposition peut être trouvé dans une note de Bernstein [Berc].

Dans la littérature du calcul formel, c'est surtout son caractère de théorème d'existence qui est exploité : connaissant un algorithme pour une certaine opération linéaire, on en déduit l'existence d'un algorithme de même complexité pour l'opération duale. Un fait mis en évidence plus récemment [BLS03] est que la transposition des programmes peut se faire de manière systématique et (quasi-)automatique. Les algorithmes sont transposés directement, d'une manière similaire à celle utilisée en différentiation automatique [GLM91], mais en tirant profit des spécificités linéaires. Par ailleurs, lorsqu'un problème linéaire doit être résolu efficacement, une démarche naturelle consiste à essayer de trouver un algorithme rapide résolvant le problème dual. Le cas échéant, une solution rapide pour le problème de départ est obtenue en re-transposant cet algorithme. C'est le point de vue adopté en Section 12.4.

Produit médian

Le concept de *produit médian* a été introduit par Hanrot, Quercia et Zimmermann en 2004 [HQZ04] : ils ont souligné l'importance de cette *nouvelle opération* sur les polynômes. Néanmoins, en traitement du signal [Blu68 ; CG00], il était déjà connu

1. Une telle famille de polynômes est appelée *suite de Sheffer* [Rom84].

que, si la multiplication est effectuée à base de DFT, le produit médian en degrés $(n, 2n)$ a la même complexité que le produit en degré n. En effet, il se code matriciellement en un produit matrice-vecteur par une matrice de Hankel de taille n ; or, celle-ci peut être plongée dans une matrice circulante de taille $2n$. Cela implique que dans le modèle de multiplication polynomiale par FFT, un produit médian $(n, 2n)$ peut être effectué en utilisant 3 DFT (deux directes et une inverse) de taille uniquement $2n$ (au lieu de $3n$). Cette interprétation n'est plus utile dans le modèle de multiplication par l'algorithme de Karatsuba, et Hanrot, Quercia et Zimmermann [HQZ04] sont donc les premiers à avoir proposé un algorithme de type Karatsuba pour le produit médian.

Lien avec la différentiation automatique

Il a été remarqué par Canny, Kaltofen, et Yagati [CKY89 ; Kal93a] que le principe de transposition est lié au *mode inverse* en *différentiation automatique* pour le calcul du gradient d'une fonction. En effet, les éléments de ${}^t\mathbf{M} \cdot \mathbf{w}$ sont les dérivées partielles de ${}^t\mathbf{v} \cdot {}^t\mathbf{M} \cdot \mathbf{w}$ par rapport aux coordonnées de $\mathbf{v}$. Ce produit n'est autre que ${}^t\mathbf{w} \cdot \mathbf{M} \cdot \mathbf{v}$, et le théorème de Baur–Strassen [BS83] montre que l'on peut calculer ces dérivées partielles au prix d'un surcoût linéaire.

Lien avec la théorie de la complexité bilinéaire

Hopcroft et Musinski ont donné une *version bilinéaire* de l'algorithme de Tellegen [HM73] : étant donné un algorithme bilinéaire qui utilise N multiplications pour le problème, noté (m, n, p), de la multiplication de deux matrices de tailles $m \times n$ et $n \times p$, il est possible d'engendrer cinq algorithmes duaux, chacun utilisant exactement N multiplications, pour les problèmes (n, m, p), (p, m, n), (m, p, n), (n, p, m) et (p, n, m). Ce résultat admet la conséquence utile suivante (mentionnée en page 183) : si l'on peut calculer en N multiplications dans $\mathbb{K}$ le produit de deux matrices quelconques sur $\mathbb{K}$ de tailles $m \times n$ et $n \times p$, alors $\omega \leq 3 \log_{mnp} \mathrm{N}$.

Dualité pour l'évaluation des monômes à plusieurs variables

Knuth et Papadimitriou [KP81] ont montré que si $\mathbf{M} = (a_{i,j})$ est une matrice carrée inversible dont les éléments sont des entiers strictement positifs, alors, partant de $\{\mathrm{X}_1, \ldots, \mathrm{X}_n\}$, le nombre minimum de multiplications pour évaluer les monômes

$$\left\{ \mathrm{X}_1^{a_{11}} \mathrm{X}_2^{a_{12}} \cdots \mathrm{X}_n^{a_{1n}},\ \mathrm{X}_1^{a_{21}} \mathrm{X}_2^{a_{22}} \cdots \mathrm{X}_n^{a_{2n}}, \ldots,\ \mathrm{X}_1^{a_{n1}} \mathrm{X}_2^{a_{n2}} \cdots \mathrm{X}_n^{a_{nn}} \right\}$$

est le même que le nombre minimum de multiplications pour évaluer les monômes

$$\left\{ \mathrm{X}_1^{a_{11}} \mathrm{X}_2^{a_{21}} \cdots \mathrm{X}_n^{a_{n1}},\ \mathrm{X}_1^{a_{12}} \mathrm{X}_2^{a_{22}} \cdots \mathrm{X}_n^{a_{n2}}, \ldots,\ \mathrm{X}_1^{a_{1n}} \mathrm{X}_2^{a_{2n}} \cdots \mathrm{X}_n^{a_{nn}} \right\}.$$

Cet énoncé peut être vu comme cas particulier du théorème de Tellegen.

Lien entre les algorithmes de Keller-Gehrig et de Storjohann

L'itération de Keller-Gehrig (Chapitre 8, page 176) permettant le calcul de la suite

$$\mathsf{s} = [x, ax, a^2x, \ldots, a^{\mathrm{N}}x]$$

$(\mathrm{N} = 2^k - 1)$ est la transposée de l'algorithme « diviser pour régner » pour d'évaluation d'un polynôme $\mathrm{P}(x) = a_0 + \cdots + a_{\mathrm{N}} x^{\mathrm{N}}$ en a via la décomposition de type *decimation-in-time* $\mathrm{P}(x) = \mathrm{P}_0(x^2) + x \cdot \mathrm{P}_1(x^2)$. L'algorithme « diviser pour régner » pour l'évaluation de

P en a via la décomposition de type *decimation-in-frequency* $P(x) = P_0(x) + x^{\frac{N+1}{2}} P_1(x)$ admet comme dual l'algorithme de Storjohann (Chapitre 11, page 212) pour calculer s.

Bibliographie

Ant79 Antoniou, A. (1979). *Digital filters : analysis and design*. McGraw-Hill Book Co.

Berc Bernstein, Daniel J. (2006). *The transposition principle*. url : http://cr.yp.to/transposition.html (visible en 2006).

BLS03 Bostan, Alin, Grégoire Lecerf et Éric Schost (2003). « Tellegen's principle into practice ». In : *ISSAC'03 : International Symposium on Symbolic and Algebraic Computation*. Éd. par J. R. Sendra. ACM Press, p. 37–44.

Blu68 Bluestein, Leo I. (1968). « A linear filtering approach to the computation of the discrete Fourier transform ». In : *IEEE Northeast Electronics Research and Engineering Meeting*, vol. 10, p. 218–219.

Bor56 Bordewijk, J. L. (1956). « Inter-reciprocity applied to electrical networks ». In : *Applied Scientific Research, Section B*, vol. 6, p. 1–74.

Bos+04a Bostan, A., G. Lecerf, B. Salvy, É. Schost et B. Wiebelt (2004). « Complexity issues in bivariate polynomial factorization ». In : *ISSAC'04 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 42–49.

BS04 Bostan, A. et É. Schost (2004). « On the complexities of multipoint evaluation and interpolation ». In : *Theoretical Computer Science*, vol. 329, n°1–3, p. 223–235.

BS83 Baur, W. et V. Strassen (1983). « The complexity of partial derivatives ». In : *Theoretical Computer Science*, vol. 22, p. 317–330.

BSS08 Bostan, Alin, Bruno Salvy et Éric Schost (2008). « Power series composition and change of basis ». In : *ISSAC'08 : International Symposium on Symbolic and Algebraic Computation*. Éd. par David J. Jeffrey. ACM Press, p. 269–276.

BT88 Ben-Or, M. et P. Tiwari (1988). « A deterministic algorithm for sparse multivariate polynomial interpolation ». In : *STOC'88 : ACM Symposium on Theory of Computing*. ACM Press, p. 301–309.

CG00 Chu, Eleanor et Alan George (2000). *Inside the FFT black box*. Serial and parallel fast Fourier transform algorithms. CRC Press.

CKY89 Canny, John F., Erich Kaltofen et Lakshman Yagati (1989). « Solving systems of non-linear polynomial equations faster ». In : *ISSAC'89 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 121–128.

CMP87 Cerlienco, L., M. Mignotte et F. Piras (1987). « Suites récurrentes linéaires. Propriétés algébriques et arithmétiques ». In : *L'Enseignement Mathématique*. II, vol. 33, p. 67–108.

Fet71 Fettweis, A. (1971). « A general theorem for signal-flow networks, with applications ». In : *Archiv für Elektronik und Übertragungstechnik*, vol. 25, n°12, p. 557–561.

Fid72a Fiduccia, C. M. (1972). « On obtaining upper bounds on the complexity of matrix multiplication ». In : *Complexity of computer computations*. IBM Thomas J. Watson Research Center, Yorktown Heights, New York : Plenum, p. 31–40.

Fid73 — (1973). « On the algebraic complexity of matrix multiplication ». Thèse de doctorat. Brown Univ., Providence, RI, Center Comput. Inform. Sci., Div. Engin.

GG99 Gathen, Joachim von zur et Jürgen Gerhard (1999). *Modern computer algebra*. Cambridge University Press.

GLM91 Gilbert, J.-C., G. Le Vey et J. Masse (1991). *La différentiation automatique de fonctions représentées par des programmes*. Rapp. tech. RR INRIA 1557.

HM73 Hopcroft, J. et J. Musinski (1973). « Duality applied to the complexity of matrix multiplication and other bilinear forms ». In : *SIAM Journal on Computing*, vol. 2, p. 159–173.

HQZ04 Hanrot, Guillaume, Michel Quercia et Paul Zimmermann (2004). « The middle product algorithm I ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 14, n°6, p. 415–438.

Kal59 Kalman, R. E. (1959). « On the general theory of control systems ». In : *IRE Transactions on Automatic Control*, vol. 4, n°3, p. 481–491.

Kal93a Kaltofen, Erich (1993). « Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems ». In : *Applied algebra, algebraic algorithms and error-correcting codes*. Vol. 673. Lecture Notes in Computer Science. Springer-Verlag, p. 195–212.

Kal93b — (1993). « Computational differentiation and algebraic complexity theory ». In : *Workshop Report on First Theory Institute on Computational Differentiation*, p. 28–30.

KCJ00 Kaltofen, E., R. M. Corless et D. J. Jeffrey (2000). « Challenges of symbolic computation : my favorite open problems ». In : *Journal of Symbolic Computation*, vol. 29, n°6, p. 891–919.

KKB88 Kaminski, M., D. G. Kirkpatrick et N. H. Bshouty (1988). « Addition requirements for matrix and transposed matrix products ». In : *Journal of Algorithms*, vol. 9, n°3, p. 354–364.

KP81 Knuth, Donald E. et Christos H. Papadimitriou (1981). « Duality in addition chains ». In : *Bulletin of the European Association for Theoretical Computer Science*, vol. 13, p. 2–4.

LS03 Lecerf, G. et É. Schost (2003). « Fast multivariate power series multiplication in characteristic zero ». In : *SADIO Electronic Journal on Informatics and Operations Research*, vol. 5, n°1, p. 1–10.

PSD70 Penfield Jr., P., R. Spence et S. Duinker (1970). *Tellegen's theorem and electrical networks*. The M.I.T. Press, Cambridge, Mass.-London.

Rom84 Roman, Steven (1984). *The umbral calculus*. Vol. 111. Pure and Applied Mathematics. Academic Press Inc. [Harcourt Brace Jovanovich Publishers].

Sho91a Shoup, V. (1991). « A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic ». In : *ISSAC'91 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 14–21.

Sho94 Shoup, Victor (1994). « Fast construction of irreducible polynomials over finite fields ». In : *Journal of Symbolic Computation*, vol. 17, n°5, p. 371–391.

Sho95 Shoup, V. (1995). « A new polynomial factorization algorithm and its implementation ». In : *Journal of Symbolic Computation*, vol. 20, n°4, p. 363–397.

Sho99 — (1999). « Efficient computation of minimal polynomials in algebraic extensions of finite fields ». In : *ISSAC'99 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 53–58.

Str73 Strassen, V. (1972/73). « Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten ». In : *Numerische Mathematik*, vol. 20, p. 238–251.

Tel52 Tellegen, B. (1952). « A general network theorem, with applications ». In : *Philips Research Reports*, vol. 7, p. 259–269.

Wie86 Wiedemann, D. (1986). « Solving sparse linear equations over finite fields ». In : *IEEE Transactions on Information Theory*, vol. IT-32, p. 54–62.

Zip90 Zippel, R. (1990). « Interpolating polynomials from their values ». In : *Journal of Symbolic Computation*, vol. 9, n°3, p. 375–403.

13. Équations différentielles à coefficients séries

Résumé

L'algorithme de Newton dans un cadre différentiel permet de calculer des séries tronquées solutions d'équations différentielles en complexité quasi-optimale. Pour certaines classes particulières importantes d'équations, la complexité peut être encore abaissée.

Comme pour le Chapitre 3 sur les calculs rapides de séries, N sera utilisé pour représenter le nombre de termes à calculer, et « série » sera employé pour « série tronquée à l'ordre N ». L'efficacité des algorithmes sera mesurée par leurs complexités arithmétiques. Nous nous attacherons à expliciter la dépendance de la complexité vis-à-vis des autres paramètres (ordre r de l'équation, degré d des coefficients lorsqu'ils sont polynomiaux). Pour le cas particulier très important des équations et des systèmes différentiels *linéaires*, les résultats de complexité de ce chapitre sont présentés dans la Table 13.1, dans laquelle il faut comparer les complexités aux tailles des entrées et des sorties. Par comparaison, la méthode la plus directe (les coefficients indéterminés), est quadratique en N pour le cas où les coefficients sont des séries.

Tous les algorithmes sont énoncés pour des coefficients dans un anneau $\mathbb{A}$ (avec en vue le cas de coefficients polynomiaux par exemple). *Dans tout ce chapitre, nous ferons l'hypothèse supplémentaire que* $2, 3, \ldots, \mathrm{N}$ *sont inversibles dans* $\mathbb{A}$. Pour les cas très importants d'équations différentielles linéaires à coefficients polynomiaux ou constants (et non pas séries), des algorithmes encore plus rapides sont présentés en fin de chapitre.

13.1 Équations différentielles d'ordre 1

Le cas de l'ordre 1 est plus simple que le cas général. Sa présentation sert d'introduction à la section suivante, où les techniques employées ici seront généralisées.

Problème (entrée, sortie)	coefficients séries	coefficients polynômes	coefficients constants	taille résultat
(équation, base)	$O(r^2\mathsf{M}(\mathrm{N}))$	$O(dr^2\mathrm{N})$	$O(r\mathrm{N})$	$r\mathrm{N}$
(équation, 1 solution)	$O(r\,\mathsf{M}(\mathrm{N})\log\mathrm{N})$	$O(dr\mathrm{N})$	$O\left(\mathsf{M}(r)\frac{\mathrm{N}}{r}\right)$	N
(système, base)	$O(r^2\mathsf{M}(\mathrm{N}))$	$O(dr^\theta\mathrm{N})$	$O(r\mathsf{M}(r)\mathrm{N})$	$r^2\mathrm{N}$
(système, 1 solution)	$O(r^2\,\mathsf{M}(\mathrm{N})\log\mathrm{N})$	$O(dr^2\mathrm{N})$	$O(\mathsf{M}(r)\mathrm{N})$	$r\mathrm{N}$

Table 13.1 – Complexité de la résolution d'équations et de systèmes différentiels linéaires pour $\mathrm{N} \gg r$.

L'équation linéaire homogène du premier ordre

Cette équation,

$$y' = \mathrm{A}(\mathrm{X})y,$$

où $\mathrm{A}(\mathrm{X})$ est une série, admet la solution

$$y(\mathrm{X}) = y(0)\exp\left(\int \mathrm{A}(\mathrm{X})\right). \tag{13.1}$$

Le calcul utilisant cette formule est dominé par celui de l'exponentielle, donc en $O(\mathsf{M}(\mathrm{N}))$ par les algorithmes du Chapitre 3.

L'équation linéaire inhomogène du premier ordre

Il s'agit de l'équation $y' = \mathrm{A}(\mathrm{X})y + \mathrm{B}(\mathrm{X})$, où A et B sont des séries. La méthode de la variation de la constante consiste à poser

$$y(\mathrm{X}) = f(\mathrm{X})\exp\left(\int \mathrm{A}(\mathrm{X})\right),$$

et à injecter cette expression dans l'équation pour obtenir que f vérifie

$$f'(\mathrm{X}) = \mathrm{B}(\mathrm{X})\exp\left(-\int \mathrm{A}(\mathrm{X})\right).$$

L'ensemble du calcul de y est donc encore borné par $O(\mathsf{M}(\mathrm{N}))$ opérations.

Le cas non linéaire

On considère maintenant des équations non linéaires de la forme $y' = \mathrm{F}(\mathrm{X}, y)$, réduites dans le résultat suivant au cas linéaire ci-dessus.

Proposition 13.1 Soit $F(X,Y) \in \mathbb{A}[[X,Y]]$ une série à deux variables telle que, pour toute série $s(X) \in \mathbb{A}[[X]]$, les N premiers termes de $F(X,s(X))$ et de $\frac{\partial F}{\partial y}(X,s(X))$ se calculent en $O(L(N))$ opérations. Alors, l'équation différentielle

$$y' = F(X,y), \qquad y(0) = a,$$

admet une série formelle solution, et ses N premiers termes peuvent être calculés en $O(L(N) + M(N))$ opérations.

L'énoncé de cette proposition en termes d'une fonction L permet de l'adapter à différents besoins : dans le cas le pire, il est possible d'invoquer l'algorithme de Brent–Kung du Chapitre 3 pour avoir $L(N) = O(N\sqrt{N \log N}\, M(N))$ qui domine alors la complexité de la résolution. Cependant, pour des équations plus simples, la composition avec F et sa dérivée pourra s'effectuer en $O(M(N))$. Ce sera le cas par exemple si F s'obtient à l'aide d'un nombre constant de sommes, d'exponentielles, de logarithmes, et de puissances.

Démonstration. L'idée de la preuve repose sur la méthode de Newton, appliquée cette fois-ci à un opérateur :

$$\Phi : y \mapsto y' - F(X,y).$$

Le principe consiste à linéariser l'opérateur Φ au voisinage d'une approximation et à en annuler la partie linéaire, pour obtenir une nouvelle approximation. À partir de

$$\Phi(y+h) = \Phi(y) + h' - \frac{\partial F}{\partial y}(X,y)h + O(h^2),$$

il s'agit donc de calculer un incrément h solution de l'équation *linéaire* inhomogène du premier ordre

$$h' = \frac{\partial F}{\partial y}(X,y)h - \Phi(y),$$

qui peut être résolue par la méthode de la section précédente. On déduit l'itération suivante

$$y_{k+1} := y_k + h_k, \quad h_k := \exp\left(\int \frac{\partial F}{\partial y}(X,y_k)\right)\int \left(F(X,y_k) - y_k'\right)\exp\left(-\int \frac{\partial F}{\partial y}(X,y_k)\right) \bmod X^{2^{k+1}}.$$

Il reste à prouver la convergence quadratique. Comme pour le théorème sur l'itération de Newton sur les séries du Chapitre 3 (page 64), la preuve se fait par récurrence sur k en prouvant simultanément $\Phi(y_k) = O(X^{2^k})$ et $h_k = O(X^{2^k})$. Les calculs sont les mêmes et sont donc laissés en exercice.

Pour obtenir le résultat de complexité, il suffit d'observer que l'itération requiert à chaque étape la résolution d'une équation différentielle linéaire inhomogène du premier ordre et d'invoquer le théorème « diviser pour régner ». ■

13.2 Équations différentielles linéaires d'ordre supérieur et systèmes d'ordre 1

L'équation différentielle d'ordre r

$$a_r(\mathrm{X})y^{(r)}(\mathrm{X}) + \cdots + a_1(\mathrm{X})y'(\mathrm{X}) + a_0(\mathrm{X})y(\mathrm{X}) = 0 \tag{13.2}$$

où les coefficients a_i sont des séries en X, avec $a_r(0) \neq 0$, peut être récrite comme une équation d'ordre 1

$$\mathrm{Y}' = \mathrm{A}(\mathrm{X})\mathrm{Y} \tag{13.3}$$

en prenant pour Y le vecteur de séries $(y(\mathrm{X}), \ldots, y^{(r-1)}(\mathrm{X}))$ et A une matrice (compagnon) de séries en X.

À l'inverse, un système (13.3) induit une relation de dépendance linéaire à coefficients des séries entre $\mathrm{Y}, \mathrm{Y}', \mathrm{Y}'', \ldots, \mathrm{Y}^{(r)}$ si r est la dimension de A (on a $r+1$ vecteurs en dimension r). Le vecteur Y et chacun de ses coefficients vérifient donc une équation de type (13.2).

Résoudre un problème est donc équivalent à résoudre l'autre. Dans certains cas, exploiter le caractère compagnon de la matrice induite par (13.2) mène à une meilleure complexité pour le cas des équations.

Une différence importante avec le cas des équations d'ordre 1 est que nous avons, pour les équations de type (13.2) comme pour les systèmes (13.3) deux problèmes à considérer :

— calculer une base des séries solutions ;

— étant données des conditions initiales, calculer *la* série solution correspondante.

Les complexités de ces problèmes sont liées : si l'on sait résoudre le second pour un coût C, alors on sait résoudre le premier pour un coût borné par $r\mathrm{C}$. Si l'on sait résoudre le premier, alors une combinaison linéaire des solutions permet de résoudre le second en au plus $r\mathrm{N}$ opérations supplémentaires.

Il est cependant utile de considérer les quatre problèmes (système ou équation, base ou solution unique) car la structure de chacun des problèmes permet de concevoir des algorithmes plus efficaces que n'en donnent les conversions directes d'un problème à l'autre.

Une méthode par « diviser pour régner »

L'équation à résoudre à précision X^{N} est ici

$$\mathrm{Y}' - \mathrm{AY} = \mathrm{B}, \quad \mathrm{Y}(0) = v,$$

où A est une matrice de séries formelles. L'idée est de résoudre d'abord à précision moitié et de déterminer puis résoudre l'équation satisfaite par le reste. La condition initiale est d'abord traitée séparément en écrivant $\mathrm{Y} = v + \mathrm{XU}$ et en l'injectant dans l'équation, ce qui mène à

$$\mathrm{XU}' + (\mathrm{I} - \mathrm{XA}(\mathrm{X}))\mathrm{U} = \mathrm{C}, \quad \mathrm{C} = \mathrm{B} + \mathrm{A}(\mathrm{X})v.$$

Soient maintenant $d < \mathrm{N}$ et $\mathrm{U} = \mathrm{U}_0 + \mathrm{X}^d\mathrm{U}_1$ où U_0 est un polynôme de degré au plus $d-1$ en X, l'équation satisfaite par U donne :

$$\mathrm{XU}_1' + ((d+1)\mathrm{I} - \mathrm{XA}(\mathrm{X}))\mathrm{U}_1 = -\mathrm{X}^{-d}(\mathrm{XU}_0' + (\mathrm{I} - \mathrm{XA}(\mathrm{X}))\mathrm{U}_0 - \mathrm{C}).$$

Entrée $A_0,\dots,A_{N-p}$ dans $\mathcal{M}_{r\times r}(\mathbb{A})$, $A=\sum A_i X^i$,
$s_0,\dots,s_{N-p}$ dans $\mathcal{M}_{r\times \ell}(\mathbb{A})$, $s=\sum s_i X^i$, $p\in\{1,\dots,N\}$.
Sortie y dans $\mathcal{M}_{r\times\ell}(\mathbb{A})[X]$ tel que $\deg_X y \le N-p$,
et $Xy'+(pI-XA)y = s+O(X^{N-p+1})$.

1. Initialiser m à $N-p$.
2. Si $m=1$ alors renvoyer $p^{-1}s(0)$ sinon :
   a. Faire $d=\lfloor m/2\rfloor$.
   b. Appeler récursivement l'algorithme avec les entrées A, $p+d$, $s \bmod X^d$, et mettre le résultat dans y_0.
   c. $R=\left[s-Xy_0'-(pI-XA)y_0\right]_d^m / X^d$.
   d. Appeler récursivement l'algorithme avec les entrées A, $p+d$, R, et mettre le résultat dans y_1.
   e. Renvoyer $y_0+X^d y_1$.

Algorithme 13.1 – Solution approchée de $XY'+(pI-XA)Y=s$.

Si U_0 est une solution à précision d, le membre droit de l'équation est bien un polynôme. L'application de la méthodologie « diviser pour régner » amène donc naturellement à considérer l'équation plus générale

$$XY'+(pI-XA)Y=R,$$

où R est une série, A une matrice de séries et $p\in\{1,\dots,N\}$. Celà nous conduit à l'Algorithme 13.1, où la notation $[s]_d^m$ désigne le polynôme obtenu en ne conservant que les termes de s de degrés au moins d et au plus $m-1$. La suite de la résolution est détaillée dans l'Algorithme 13.2. Les résultats de complexité se déduisent de cette méthode pour différentes valeurs de A et de ℓ. En particulier, le cas des équations a une meilleure complexité que le cas général d'un système d'ordre 1 car la matrice correspondante est une matrice compagnon. Le produit d'une telle matrice par un vecteur ne coûte que r opérations au lieu de r^2 dans le cas général.

Théorème 13.2 — Diviser pour régner pour les équations différentielles.
Étant donnés les N premiers coefficients de $A\in\mathcal{M}_{r\times r}(\mathbb{A}[[X]])$, de $B\in\mathcal{M}_{r\times\ell}(\mathbb{A}[[X]])$ ainsi que des conditions initiales $v\in\mathcal{M}_{r\times\ell}(\mathbb{A})$, l'Algorithme 13.2 calcule l'unique solution de

$$Y'-AY=B+O(X^N),\qquad Y(0)=v$$

en un nombre d'opérations dans $\mathbb{A}$ borné par

1. $O(r^2\ell M(N)\log N)$ en général ;
2. $O(r\ell M(N)\log N)$ si A est une matrice compagnon.

Il est possible d'améliorer les estimations de complexité pour cet algorithme pour la valeur $\ell=r$ (calcul de toute une base des solutions), mais dans ce cas, il vaut mieux employer les algorithmes de la section suivante, qui sont plus efficaces.

Démonstration. Il suffit de prouver le résultat pour N une puissance de 2, et le cas

Entrée $A_0, \dots, A_N$ dans $\mathcal{M}_{r\times r}(\mathbb{A})$, $A = \sum A_i X^i$,
$B_0, \dots, B_N$, et v dans $\mathcal{M}_{r\times \ell}(\mathbb{A})$, $B = \sum B_i X^i$.
Sortie $y = \sum_{i=0}^{N} y_i X^i$ dans $\mathcal{M}_{r\times \ell}(\mathbb{A})[X]$ tel que
$y' - Ay = B$ et $y(0) = v$.

1. Appeler l'Algorithme 13.1 sur A, 1, et $B + Av$, et mettre le résultat dans y.
2. Renvoyer $v + Xy$.

Algorithme 13.2 – Solution approchée de $Y' - AY = B$.

général s'en déduit par les hypothèses habituelles sur la fonction de multiplication M.

Les opérations de troncature ne demandent pas de calcul dans $\mathbb{A}$. Outre les deux appels récursifs, le calcul demande une dérivation (linéaire), un produit par p fois l'identité (linéaire) et un produit par A. La complexité C(N) vérifie donc

$$C(N) = 2C(N/2) + \lambda \ell N + \mathsf{Mult}(A, V, N),$$

où $\mathsf{Mult}(A, V, N)$ désigne le coût du produit de la matrice A par la matrice $r \times \ell$ de séries à précision N. La conclusion s'obtient par le théorème « diviser pour régner » en observant les complexités de base pour $\mathsf{Mult}(A, V, N)$. ■

L'itération de Newton matricielle

Comme dans la preuve de la Proposition 13.1, le point de départ de l'algorithme consiste à appliquer l'itération de Newton à l'opérateur dont on cherche les solutions, en linéarisant au voisinage d'une solution approchée. L'opérateur

$$Y \mapsto Y' - A(X)Y$$

étant linéaire, il est son propre linéarisé. Formellement, l'itération de Newton s'écrit donc

$$Y_{k+1} = Y_k - U_k, \quad U_k' - AU_k = Y_k' - AY_k.$$

Lorsque Y_k est une solution approchée, le membre droit de l'équation en U_k a une valuation strictement positive, et la solution U_k cherchée doit avoir aussi une telle valuation. Une telle solution est obtenue par la méthode de la variation de la constante *si l'on dispose d'une base des solutions*, c'est-à-dire dans notre cas si Y_k est une matrice $r \times r$ avec $\det Y_k(0) \neq 0$. Dans ce cas, la méthode de la variation de la constante suggère de poser $U_k = Y_k T$, ce qui donne

$$U_k' - AU_k = Y_k T' + \underbrace{AY_k T - AY_k T}_{0} = Y_k' - AY_k$$

d'où finalement

$$U_k = Y_k \int Y_k^{-1}(Y_k' - AY_k).$$

Entrée $Y_0, A_0, \dots, A_{N-2}$ dans $\mathcal{M}_{r\times r}(\mathbb{A})$, $A = \sum A_i X^i$.
Sortie $Y = \sum_{i=0}^{N-1} Y_i X^i$ dans $\mathcal{M}_{r\times r}(\mathbb{A})[X]$ tel que
$Y' = AY + O(X^{N-1})$, et $Z = Y^{-1} + O(X^{N/2})$.

1. $Y = (I + XA_0)Y_0$.
2. $Z = Y_0^{-1}$.
3. $m = 2$.
4. Tant que $m \le N/2$ faire :
   $Z = Z + Z(I_r - YZ) \bmod X^m$;
   $Y = Y - Y\left(\int Z(Y' - A \bmod X^{2m-1} Y)\right) \bmod X^{2m}$;
   $m = 2m$.
5. Renvoyer Y, Z.

Algorithme 13.3 – Résolution de $Y' = A(X)Y$, $Y(0) = Y_0$ par itération de Newton.

Cette méthode requiert le calcul de l'inverse d'une base de solution, inverse qui peut être calculé simultanément par l'itération de Newton :

$$Z_{k+1} = Z_k + Z_k(I - Y_k Z_k).$$

Une version précise de la méthode, avec les ordres de troncature, est donnée dans l'Algorithme 13.3.

Lemme 13.3 — Correction de l'algorithme. Soit m un entier pair, soient y et z dans $\mathcal{M}_{r\times r}(\mathbb{A}[X])$ tels que

$$I - yz = O(X^{m/2}), \quad y' - Ay = O(X^m).$$

Soient ensuite Y et Z définis par

$$Z := z(2I - yz) \bmod X^m, \quad Y := y\left(I - \int Z(y' - Ay)\right) \bmod X^{2m}.$$

Alors Y et Z vérifient

$$I - YZ = O(X^m), \quad Y' - AY = O(X^{2m-1}).$$

Démonstration. D'après l'hypothèse sur $y' - Ay = O(X^m)$, $Y = y + O(X^m)$ et donc la convergence de l'inversion est donnée par

$$\begin{aligned} I - YZ &= I - y\big(z + z(I - yz)\big) + O(X^m) \\ &= (I - yz) - yz(I - yz) + O(X^m) \\ &= (I - yz)^2 + O(X^m) = O(X^m). \end{aligned}$$

La seconde propriété s'obtient en injectant la définition de Y dans $Y' - AY$ (on pose

$Q = \int Z(y' - Ay))$:

$$\begin{aligned} Y' - AY &= y' + y'Q - Ay - AyQ - yZ(y' - Ay) + O(X^{2m}), \\ &= (I - yZ)(y' - Ay) - (y' - Ay)Q + O(X^{2m}), \\ &= O(X^m)O(X^m) + O(X^m)O(X^{m-1}) + O(X^{2m}) = O(X^{2m-1}). \end{aligned}$$ ■

Comme d'habitude, l'application du théorème « diviser pour régner » mène au résultat de complexité.

Théorème 13.4 — Newton pour les systèmes différentiels linéaires. Les N premiers termes d'une base de solutions de $Y' = AY$ sont calculés en $O(MM(r, N))$ opérations dans $\mathbb{A}$ par l'Algorithme 13.3.

La notation $MM(r, N)$ représente la complexité du produit de matrices $r \times r$ de polynômes de degré au plus N ; des bornes non triviales sont données au Chapitre 5 :

$$MM(r, N) = O(r^\theta N + r^2 M(N)).$$

Exercice 13.1 L'exponentielle d'une série est obtenue en $O(M(N))$ opérations par cet algorithme lorsque $r = 1$. Vérifier que la constante est meilleure que celle de l'algorithme du Chapitre 3. ■

Exercice 13.2 Le cas d'un système *inhomogène* $Y' = AY + B$, où B est une matrice de séries, s'obtient à partir de l'algorithme précédent par variation de la constante. Étudier les précisions requises dans les troncatures, et donner le résultat de complexité. ■

13.3 Cas particuliers

Équations différentielles linéaires à coefficients polynomiaux

Pour ces équations, la méthode des coefficients indéterminés donne une complexité linéaire en N. L'explication tient à une propriété vue au Chapitre 14 : les solutions séries de ces équations ont des coefficients qui vérifient des récurrences linéaires. Les résultats du Chapitre 15 sur les récurrences linéaires à coefficients polynomiaux s'appliquent alors, et en particulier les N premiers coefficients d'une solution s'obtiennent en $O(drN)$ opérations si d est une borne sur le degré des q_i. Le Chapitre 16 revient sur ces questions.

Le cas matriciel se traite de la même façon, la récurrence ci-dessus étant alors à coefficients matriciels. Les complexités sont de même nature, avec en outre la possibilité d'utiliser un produit rapide de matrices dans le cas du calcul d'une base de solutions. Les résultats correspondants sont donnés dans la Table 13.1.

Entrée A dans $\mathcal{M}_{r\times r}(\mathbb{A})$, $v \in \mathcal{M}_{r\times 1}(\mathbb{A})$.
Sortie $Y = \sum_{i=0}^{N-1} Y_i X^i$ dans $\mathcal{M}_{r\times r}(\mathbb{A})[X]$ tel que $Y' = AY + O(X^{N-1})$.

1. Calculer les vecteurs $v, Av, A^2v, A^3v, \ldots, A^{2r}v$.
2. Pour tout $j = 1, \ldots, r$ faire :
   a. reconstruire la fraction rationnelle z_j ayant pour développement en série $\sum (A^i v)_j X^i$;
   b. développer cette fraction en série à précision X^N ;
   c. reconstruire le développement de y_j à partir de celui de z_j.
3. Renvoyer le vecteur $(y_1, \ldots, y_r)$.

Algorithme 13.4 – Résolution approchée de $Y' = AY$.

Équations différentielles linéaires à coefficients constants

Dans le cas où les coefficients de l'équation ou du système sont constants, la formule (13.1) s'applique encore et devient :

$$y(X) = \exp(XA)y(0).$$

Pour étudier les propriétés de cette solution, il est usuel de faire intervenir la forme de Jordan de la matrice, mais cette forme ne se prête pas facilement au calcul.

Un algorithme efficace est obtenu en exploitant la *transformée de Laplace formelle*. Si $S = s_0 + s_1 X + s_2 X^2 + \cdots$ est une série, cette transformée est définie par

$$\mathcal{L}S = s_0 + 1!s_1 X + 2!s_2 X^2 + \cdots.$$

Les troncatures de $\mathcal{L}S + O(X^N)$ et de la transformée inverse $\mathcal{L}^{-1}S + O(X^N)$ se calculent en N opérations. Cette transformation simplifie les systèmes différentiels linéaires à coefficients constants en les rendant linéaires non différentiels. Ceci découle de

$$\mathcal{L}(y) = \mathcal{L}((I + XA + \frac{1}{2!}X^2A^2 + \cdots)y(0)) = (I + XA + X^2A^2 + \cdots)y(0) = (I - XA)^{-1}y(0).$$

Les vecteurs solutions ont donc des transformées de Laplace dont les coordonnées sont rationnelles. En outre, d'après les formules de Cramer, le numérateur et le dénominateur de ces fractions ont des degrés bornés par l'ordre r du système. L'Algorithme 13.4 s'en déduit. L'étape (1) est effectuée par la méthode naïve en $O(r^3)$ opérations. Une méthode plus efficace, en $O(\mathsf{MM}(r)\log r)$ opérations, est à la base de l'algorithme de Keller-Gehrig pour le calcul du polynôme caractéristique d'une matrice présenté au Chapitre 8. L'étape (2.a) se ramène à un calcul d'algèbre linéaire en posant des coefficients indéterminés pour les numérateurs et dénominateurs. Là aussi, le coût peut être diminué en faisant appel au calcul d'approximants de Padé (Chapitre 7), mais cette partie du calcul ne dépend pas de N. L'étape (2.b) utilise l'algorithme de développement de fractions rationnelles du Chapitre 4 (Théorème 4.9, page 88) en $O(N\mathsf{M}(r)/r)$ opérations, et c'est là que se concentre le gain en complexité. Enfin, l'étape (2.c) ne demande que $O(N)$ opérations. L'ensemble de l'algorithme a donc une complexité arithmétique en $O(N\mathsf{M}(r))$.

13.4 Extensions

Composer se ramène à résoudre

Le seul algorithme du Chapitre 3 dont la complexité n'est pas quasi-optimale est l'algorithme de composition.

Dans les applications où l'on connaît une équation $\Phi(X, f, f', \dots, f^{(m)}) = 0$ vérifiée par la série f il est parfois possible de calculer la série $h = f \circ g$ efficacement *sans passer par l'algorithme de composition* en remarquant qu'elle vérifie une équation du même ordre, obtenue en remplaçant X par g dans Φ et en composant les dérivées. Si Φ ne fait intervenir que des polynômes ou des fractions rationnelles, le résultat de cette opération a pour coefficients des séries, ce qui mène assez généralement à une complexité en $O(M(N))$ opérations en utilisant les algorithmes de ce chapitre.

Exemple 13.1 Pour calculer le développement de $h = \tan(f)$, il est possible de calculer en $O(M(N))$ ceux de $\sin(f)$ et $\cos(f)$ (via l'exponentielle) et de diviser, mais il est aussi possible d'observer que h vérifie l'équation $h' = 1 + h^2 f'$, et d'utiliser les méthodes ci-dessus. Il faut ensuite comparer les constantes dans les $O(\cdot)$ (ou deux implantations !) pour déterminer laquelle des deux méthodes est préférable.

Exercice 13.3 Le n^{e} polynôme de Legendre $P_n(X)$ est défini par l'équation $P_n(X) = \frac{d^n}{dX^n}\left((X^2-1)^n\right)/(2^n n!)$. Montrer qu'il vérifie l'équation différentielle

$$\frac{d}{dX}\left[(1-X^2)\frac{d}{dX}P_n(X)\right] + n(n+1)P_n(X) = 0.$$

En déduire que pour toute série de terme constant nul $F \in \mathbb{Q}[[X]]$, et pour tout $N \geq 0$, la série composée $P_N \circ F \bmod X^N$ peut se calculer en $O(M(N))$ opérations. ■

Systèmes non linéaires

La linéarisation effectuée dans le cas des équations d'ordre 1 se généralise aux systèmes. L'énoncé devient alors le suivant.

Théorème 13.5 Soient $\phi_1, \dots, \phi_r$ r séries de $\mathbb{A}[[X, Y_1, \dots, Y_r]]$, telles que pour tout r-uplet de séries $(s_1(X), \dots, s_r(X)) \in \mathbb{A}[[X]]^r$, les N premiers termes des $\phi_i(X, s_1, \dots, s_r)$ et de $\mathrm{Jac}(\phi)(X, s_1(X), \dots, s_r(X))$ puissent être calculés en utilisant $O(L(N))$ opérations dans $\mathbb{A}$. On suppose en outre que $L(n)/n$ est une suite croissante. Alors, le système différentiel

$$\begin{cases} y_1'(t) = \varphi_1(t, y_1(t), \dots, y_r(t)), \\ \quad\vdots \\ y_r'(t) = \varphi_r(t, y_1(t), \dots, y_r(t)), \end{cases}$$

avec conditions initiales $(y_1, \dots, y_r)(0) = v$ admet une solution série formelle, et ses N premiers termes peuvent être calculés en

$$O(L(N) + \min(MM(r, N), r^2 M(N) \log N)).$$

Le symbole $\mathrm{Jac}(\phi)$ désigne la matrice jacobienne : son coefficient sur la ligne i et la colonne j vaut $\partial\phi_i/\partial y_j$. L'intérêt d'énoncer le théorème à l'aide de la fonction L est le même que dans le cas des équations.

Démonstration. Il s'agit essentiellement d'une généralisation de la Proposition 13.1. La linéarisation de $\Phi : Y \mapsto Y' - \varphi(Y)$ au voisinage d'une solution approchée y s'obtient en écrivant :

$$\Phi(y+h) = \Phi(y) + h' + \mathrm{Jac}(\phi)(y)h + \mathrm{O}(h^2).$$

L'équation à résoudre pour une itération de Newton est donc le système linéaire inhomogène

$$h' = \mathrm{Jac}(\phi)(y)h - (y' - \phi(y)).$$

Si h est un vecteur solution de ce système à précision $2m$ et y une solution de Φ à précision m, ces équations montrent que $y+h$ est solution à précision $2m$. Le cas non linéaire est ainsi ramené au cas linéaire. ■

Notes

Les résultats de la Section 13.1 sont tirés de l'article de Brent et Kung de 1978 [BK78]. Une bonne partie des autres résultats de ce chapitre provient de l'article de Bostan, Chyzak, Ollivier, Salvy, Schost, et Sedoglavic de 2007 [Bos+07a]. La technique de la Section 13.2 dans le cas particulier de l'exponentielle d'une série est due à Hanrot, Quercia, et Zimmermann [HQZ04]. L'algorithme « diviser pour régner » de la Section 13.2 se trouve au moins implicitement dans des travaux de van der Hoeven [Hoe02c]. Des algorithmes plus récents peuvent être consultés dans l'article de van der Hoeven de 2010 [Hoe10].

Bibliographie

BK78 Brent, R. P. et H. T. Kung (1978). « Fast algorithms for manipulating formal power series ». In : *Journal of the ACM*, vol. 25, n°4, p. 581–595.

Bos+07a Bostan, A., F. Chyzak, F. Ollivier, B. Salvy, É. Schost et A. Sedoglavic (2007). « Fast computation of power series solutions of systems of differential equations ». In : *SODA'07 : ACM-SIAM Symposium on Discrete Algorithms*. SIAM, p. 1012–1021.

Hoe02c Hoeven, Joris van der (2002). « Relax, but don't be too lazy ». In : *Journal of Symbolic Computation*, vol. 34, n°6, p. 479–542.

Hoe10 — (2010). « Newton's method and FFT trading ». In : *Journal of Symbolic Computation*, vol. 45, n°8, p. 857–878.

HQZ04 Hanrot, Guillaume, Michel Quercia et Paul Zimmermann (2004). « The middle product algorithm I ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 14, n°6, p. 415–438.

III

Équations différentielles et récurrences linéaires

14. Séries différentiellement finies

Résumé

Les séries différentiellement finies sont les solutions d'équations différentielles linéaires à coefficients polynomiaux. Leurs coefficients vérifient des récurrences linéaires qui permettent de les manipuler plus rapidement que des séries arbitraires. L'équation différentielle linéaire ou la récurrence linéaire devient une structure de données adaptée pour manipuler les solutions. De nombreuses opérations dont l'addition et la multiplication peuvent être effectuées directement sur les équations. Une conséquence immédiate est une multiplication de ces séries différentiellement finies en temps linéaire.

Les séries différentiellement finies (c'est-à-dire solutions d'équations différentielles linéaires à coefficients polynomiaux) ont des coefficients qui satisfont une récurrence linéaire, ce qui permet d'en calculer les N premiers en O(N) opérations, donc plus vite que la plupart des autres séries. Il est de ce fait crucial de reconnaître les séries qui sont différentiellement finies et de disposer des équations différentielles les définissant. De plus, les coefficients des séries différentiellement finies forment des suites qui sont appelées polynomialement récurrentes, dont l'algorithmique est évidemment étroitement liée à celle des séries différentiellement finies. L'importance de ces séries et suites provient en grande partie de leur omniprésence dans les applications. Ainsi, le *Handbook of Mathematical Functions*, référence importante en physique, chimie et mathématiques appliquées, comporte environ 60% de fonctions solutions d'équations différentielles linéaires ; de même, les suites polynomialement récurrentes forment environ un quart des plus de 100 000 suites référencées dans la version en ligne de l'*Encyclopedia of Integer Sequences* de N. Sloane. Enfin, l'algorithmique de ces séries permet *la preuve automatique d'identités*, présentée dans ce chapitre en présence d'une seule variable. La généralisation à plusieurs variables est traitée dans les chapitres de la cinquième partie.

14.1 Équations différentielles et récurrences

Définitions

Si $\mathbb{K}$ désigne un corps, l'anneau des séries formelles à coefficients dans $\mathbb{K}$ est noté $\mathbb{K}[[X]]$. Ses principales propriétés ont été présentées au Chapitre 3. Son corps des fractions, noté $\mathbb{K}((X))$, est égal à $\mathbb{K}[[X]][1/X]$. Ses éléments sont appelés des séries de Laurent formelles. C'est une algèbre non seulement sur $\mathbb{K}$, mais aussi sur le corps des fractions rationnelles $\mathbb{K}(X)$.

Définition 14.1 Une série formelle $A(X)$ à coefficients dans un corps $\mathbb{K}$ est dite *différentiellement finie* lorsque ses dérivées successives $A, A', \ldots$, engendrent dans l'algèbre $\mathbb{K}((X))$ des séries de Laurent formelles un espace vectoriel de dimension finie sur le corps $\mathbb{K}(X)$ des fractions rationnelles.

De manière équivalente, cette série est solution d'une équation différentielle linéaire à coefficients dans $\mathbb{K}(X)$: si c'est le cas, alors l'équation différentielle permet de récrire toute dérivée d'ordre supérieur à celui de l'équation en termes des dérivées d'ordre moindre (en nombre borné par l'ordre). À l'inverse, si l'espace est de dimension finie, alors pour m suffisamment grand, $A, A', \ldots, A^{(m)}$ sont liées, et une relation de liaison entre ces dérivées est une équation différentielle linéaire.

Exemple 14.1 De nombreuses séries classiques sont de ce type : $\exp(X)$, $\log(1+X)$, $(1+X)^a$, $\sin X$, $\sinh X$, $\cos X$, $\cosh X$, $\arcsin X$, $\arccos X$, $\arctan X$,..., les séries hypergéométriques généralisées, et d'autres fonctions classiques de la physique mathématique.

R La Définition 14.1 pourrait être donnée plus généralement pour des séries de Laurent formelles de $\mathbb{K}((X))$, mais est sciemment restreinte ici aux séries formelles de $\mathbb{K}[[X]]$, dans l'objectif notamment de bien fonctionner avec la définition qui suit et avec le Théorème 14.1. Dans la Partie VI, la notion sera étendue à toute sorte d'espaces vectoriels sur $\mathbb{K}(X)$ clos par dérivation.

Définition 14.2 Une suite $(a_n)_{n\geq 0}$ d'éléments d'un corps $\mathbb{K}$ est appelée suite *polynomialement récurrente* si elle satisfait à une récurrence de la forme

$$p_d(n)a_{n+d} + p_{d-1}(n)a_{n+d-1} + \cdots + p_0(n)a_n = 0, \qquad n \geq 0, \tag{14.1}$$

où les p_i sont des polynômes de $\mathbb{K}[X]$.

Dans la suite, $\mathbb{K}$ aura toujours caractéristique nulle. On peut donc penser sans rien perdre aux idées à $\mathbb{K} = \mathbb{Q}$.

R Comme pour les séries différentiellement finies, on pourrait donner une définition en termes d'espaces vectoriels engendrés par les suites décalées, mais il faut alors faire attention aux zéros entiers du coefficient de tête de la récurrence p_d. Une possibilité est de considérer les *germes* de suites à l'infini, c'est-à-dire les classes d'équivalences de suites pour la relation « être égal à partir d'un certain

rang ». Cette approche n'est pas très effective, et nous travaillerons plutôt dans les chapitres suivants avec des conditions initiales généralisées.

Exemple 14.2 Outre les solutions de récurrences linéaires à coefficients constants comme les nombres de Fibonacci, des suites classiques de ce type sont : $n!$, $1/n!$, $\binom{2n}{n}$, les nombres harmoniques H_n, les nombres de Catalan, de Motzkin,...

Exemple 14.3 Le cas particulier des récurrences d'ordre 1 ($d = 1$) donne lieu aux suites *hypergéométriques*, qui jouent un rôle important dans les algorithmes de sommation symbolique, abordés au Chapitre 29.

Équivalence

Le résultat qu'il s'agit de considérer d'un point de vue algorithmique est le suivant. Il s'agit d'un analogue du Lemme 4.7.

Théorème 14.1 Une série formelle est différentiellement finie si et seulement si la suite de ses coefficients est polynomialement récurrente.

Démonstration. Soit $A(X) = a_0 + a_1 X + \cdots$ une série différentiellement finie et

$$q_m(X)A^{(m)}(X) + \cdots + q_0(X)A(X) = 0 \tag{14.2}$$

une équation différentielle qui l'annule. En notant $[X^n]f(X)$ le coefficient de X^n dans la série $f(X)$ avec la convention que ce coefficient est nul pour $n < 0$, on a pour $n \geq 0$

$$[X^n]f'(X) = (n+1)[X^{n+1}]f(X), \qquad [X^n]X^k f(X) = [X^{n-k}]f(X). \tag{14.3}$$

Par conséquent, l'extraction du coefficient de X^n de (14.2) fournit une récurrence linéaire sur les a_n valide pour tout $n \geq 0$ avec la convention $a_k = 0$ pour $k < 0$. Pour obtenir une récurrence de la forme (14.1), il faut décaler les indices de $n_0 := \max_{0\leq i\leq m}(\deg q_i - i)$ si cet entier est strictement positif. Les équations obtenues alors pour les indices moindres fournissent des contraintes linéaires sur les premiers coefficients a_n pour qu'ils correspondent aux coefficients d'une série solution de (14.2).

À l'inverse, soit (a_n) une suite vérifiant la récurrence (14.1). Les identités analogues à (14.3) sont maintenant

$$\sum_{n\geq 0} n^k a_n X^n = \left(X\frac{d}{dX}\right)^k A(X), \quad \sum_{n\geq 0} a_{n+k}X^n = (A(X) - a_0 - \cdots - a_{k-1}X^{k-1})/X^k,$$

où A est la série génératrice des coefficients a_n et la notation $(X\frac{d}{dX})^k$ signifie que l'opérateur $X\frac{d}{dX}$ est appliqué k fois. En multipliant (14.1) par X^n et en sommant pour n allant de 0 à ∞, puis en multipliant par une puissance de X, on obtient donc une équation différentielle linéaire de la forme

$$q_d(X)A^{(d)}(X) + \cdots + q_0(X)A(X) = p(X),$$

où le membre droit provient des conditions initiales. Il est alors possible, quitte à augmenter l'ordre de l'équation de 1, de faire disparaître ce membre droit, par une dérivation et une combinaison linéaire. ■

R Ces récritures d'équations différentielles en récurrences et réciproquement seront interprêtés comme des morphismes d'anneaux d'opérateurs dans la Partie VI.

Exemple 14.4 L'équation $A' - X^k A = 0$ ($k \in \mathbb{N}$) donne la récurrence linéaire $(n+1)a_{n+1} - a_{n-k} = 0$ valide pour tout $n \geq 0$, avec la convention que les a_n d'indice négatif sont nuls. On en déduit que a_0 est libre, puis les contraintes $a_1 = \cdots = a_k = 0$, et les coefficients suivants sont fournis par la récurrence décalée $(n+k+1)a_{n+k+1} - a_n = 0$, valide pour $n \geq 0$. On reconnaît ainsi les coefficients de $a_0 \exp(X^{k+1}/(k+1))$.

Exercice 14.1 Retrouver la récurrence satisfaite par la suite $1/n!$ à partir de l'équation $A' = A$, et réciproquement. ■

Exercice 14.2 Calculer une équation différentielle linéaire satisfaite par la série génératrice des nombres de Fibonacci, définis par $F_{n+2} = F_{n+1} + F_n$, $F_0 = 0$, $F_1 = 1$. ■

Exercice 14.3 Estimer la complexité d'un algorithme direct utilisant cette idée en nombre d'opérations dans $\mathbb{K}$. ■

Un algorithme plus efficace pour passer d'une équation différentielle d'ordre élevé à la récurrence linéaire satisfaite par les coefficients des solutions est donné en Section 14.1.

Exemples d'applications

Exemple 14.5 Pour calculer le coefficient de X^{1000} dans

$$(1+X)^{1000}(1+X+X^2)^{500},$$

une méthode efficace part de l'observation que ce polynôme vérifie l'équation différentielle linéaire du premier ordre

$$\frac{y'}{y} = 1000\frac{1}{1+X} + 500\frac{2X+1}{1+X+X^2}.$$

Il s'ensuit que les coefficients de ce polynôme vérifient une récurrence d'ordre 3 à coefficients de degré 1. Cette récurrence permet d'obtenir le N-ième coefficient en $O(N)$ opérations arithmétiques, voire $\tilde{O}(\sqrt{N})$ en utilisant la technique de « pas de bébés, pas de géants » présentée au Chapitre 15. Pour la complexité binaire, ce même chapitre présente la méthode de scindage binaire qui est quasi-optimale.

Exemple 14.6 Pour calculer une racine du polynôme $P_N(X)$ défini par la série génératrice

$$\sum_{n\geq 0} P_n(X)\frac{Z^n}{n!} = \left(\frac{1+Z}{1+Z^2}\right)^X,$$

lorsque N est grand, il n'est pas nécessaire de calculer ce polynôme. Il suffit d'observer que cette série vérifie une équation différentielle linéaire d'ordre 1 (avec X en paramètre), ainsi que la série génératrice des dérivées des P_n, et d'utiliser les récurrences que l'on en déduit sur ces polynômes pour en calculer des valeurs. Ces valeurs permettent alors d'appliquer la méthode de Newton, par exemple pour résoudre le polynôme. Cette idée peut aussi être combinée avec la méthode de scindage binaire du Chapitre 15.

Test d'égalité

Lemme 14.2 Si (u_n) et (v_n) sont deux suites solutions de (14.1), et s'il existe $k \in \mathbb{N}$ tel que $u_0 = v_0, \ldots, u_{k-1} = v_{k-1}$ et $0 \notin p_d(k-d+1+\mathbb{N})$, alors ces suites sont égales.

Démonstration. Par récurrence, puisque $0 \notin p_d(k-d+1\mathbb{N})$ permet d'inverser le coefficient de tête de la récurrence et donc d'exprimer chaque terme à partir de l'indice d en fonction des précédents. ■

Exemple 14.7 Pour une récurrence à coefficients constants, il suffit de connaître une borne sur l'ordre de la récurrence pour tester l'égalité de deux solutions. Cette observation est utile en liaison avec les opérations de clôture de la section suivante.

Corollaire 14.3 Si $f(X)$ et $g(X)$ sont deux séries formelles solutions de (14.2), $f(0) = g(0), \ldots, f^{(m-1)}(0) = g^{(m-1)}(0)$ et $q_0(0) \neq 0$, alors ces séries sont égales.

Démonstration. D'après (14.3), un terme $cX^iA^{(j)}(X)$ intervient dans la récurrence sur les coefficients sous la forme $c(n-i+1)\cdots(n-i+j)a_{n-i+j}$. L'indice maximal est donc atteint pour $j-i$ maximal et donc pour $j=m$ si $i=0$. La récurrence obtenue, valide pour $n \geq 0$ est donc de la forme $q_0(0)(n+1)\cdots(n+m)a_{n+m} + \cdots = 0$. Les contraintes linéaires sur les conditions initiales jusqu'à l'indice n_0 introduit dans la preuve du Théorème 14.1 définissent les coefficients d'indice m à $m+n_0$ à partir des précédents et le lemme précédent s'applique pour les valeurs suivantes. ■

Algorithme rapide pour la conversion

Vue l'efficacité obtenue grâce au passage de l'équation différentielle à la récurrence, il est souhaitable de maîtriser le coût de cette conversion. Il est possible d'obtenir la récurrence plus efficacement que par la méthode naïve, en mettant en œuvre des techniques qui exploitent la multiplication rapide.

Cas d'un opérateur donné en $\theta = X\frac{d}{dX}$

Si l'équation différentielle linéaire de départ est donnée non pas comme un polynôme en X et d/dX, mais comme un polynôme en X et $\theta = X\frac{d}{dX}$ (θ est parfois appelé *opérateur d'Euler*), alors la conversion en récurrence est assez facile : l'application de l'opérateur

$$\sum_{\substack{0\le j\le m\\ 0\le i\le d}} p_{i,j} X^j \theta^i,$$

à une série A(X) suivie de l'extraction du coefficient de X^n donne au vu des relations (14.3) la récurrence

$$\sum p_{i,j}(n-j)^i a_{n-j} = 0.$$

De cette conversion découle le résultat de complexité suivant.

Proposition 14.4 La récurrence satisfaite par les coefficients des séries solutions d'une équation différentielle linéaire de degré d en $\theta = X\frac{d}{dX}$ et m en X se calcule en $O(m\mathsf{M}(d))$ opérations sur les coefficients.

Par rapport au nombre dm de coefficients du résultat, cette complexité est quasi-optimale.

La preuve utilise la formule ci-dessus et l'observation que calculer les coefficients du polynôme $P(X-j)$ connaissant ceux du polynôme P(X) de degré d ne requiert que $O(\mathsf{M}(d))$ opérations, par exemple en utilisant la somme composée (Chapitre 3).

Cas général

Quitte à le multiplier au préalable par une puissance de X égale au plus à son degré en $\partial = d/dX$, il est toujours possible de récrire un polynôme en X et ∂ en un polynôme en X et θ. Cette récriture peut elle-même être effectuée assez rapidement.

Une première observation est que de la *commutation*

$$(\theta - i)X^i = X^i\theta$$

se déduit par multiplication à droite par ∂^i la relation

$$X^{i+1}\partial^{i+1} = (\theta - i)X^i\partial^i = (\theta - i)(\theta - i + 1)\cdots\theta.$$

Étant donnés des polynômes $p_i(X)$ de degré au plus $m+d$, il s'agit donc maintenant de calculer des polynômes $q_i(X)$ tels que

$$\sum_{i=0}^{d} p_i(X)X^i\partial^i = \sum_{i=0}^{d} p_i(X)(\theta - i + 1)\cdots\theta = \sum_{i=0}^{d} q_i(X)\theta^i.$$

Récrire le polynôme sous la forme

$$\sum_{j=0}^{m+d} X^j \sum_{i=0}^{d} p_{i,j} X^i \partial^i$$

s'effectue en nombre linéaire d'opérations et montre qu'il suffit de savoir traiter efficacement le cas où les p_i (et donc aussi les q_i) sont constants. La transition des uns vers les autres se calcule alors par évaluation-interpolation sur $\theta = 0, 1, 2, \ldots$. Soit P le polynôme à calculer. Les premières identités obtenues par évaluation sont

$$p_0 = q_0, \quad p_0 + p_1 = \sum q_i, \quad p_0 + 2p_1 + 2p_2 = \sum 2^i q_i,$$

et plus généralement

$$e^X \sum p_i X^i = \sum \frac{P(i)}{i!} X^i,$$

ce qui montre que les valeurs de P en $0, \ldots, d$ peuvent être obtenues en $O(M(d))$ opérations à partir des coefficients p_i, et donc les coefficients de P en $O(M(d)\log d)$ opérations, par interpolation rapide, en utilisant les techniques du Chapitre 5. En résumé, nous avons obtenu le résultat suivant.

Théorème 14.5 Le calcul de la récurrence linéaire satisfaite par les coefficients des séries solution d'une équation différentielle linéaire d'ordre d à coefficients des polynômes de degré au plus m requiert un nombre d'opérations arithmétiques borné par

$$O((m+d)M(d)\log d).$$

14.2 Propriétés de clôture

Les séries différentiellement finies jouissent de propriétés de clôture. En ce sens, elles sont un analogue différentiel des nombres algébriques, et une généralisation des suites récurrentes linéaires à coefficients *constants* (srlcc) évoquées en Section 4.5.

Somme et produit

Théorème 14.6 L'ensemble des séries différentiellement finies à coefficients dans un corps $\mathbb{K}$ est une algèbre sur $\mathbb{K}$. L'ensemble des suites polynomialement récurrentes d'éléments de $\mathbb{K}$ est aussi une algèbre sur $\mathbb{K}$.

Une conséquence algorithmique fondamentale de ce théorème est le fait que les séries différentiellement finies peuvent être multipliées plus rapidement que les séries arbitraires, en complexité *linéaire*.

Corollaire 14.7 Soient $A, B \in \mathbb{K}[[X]]$ deux séries différentiellement finies, solutions d'équations différentielles d'ordre $O(1)$ à coefficients polynomiaux de degré $O(1)$. On peut calculer le produit AB mod X^N en $O(N)$ opérations dans $\mathbb{K}$.

Démonstration du théorème. Les preuves pour les suites et les séries sont similaires. Les preuves pour les sommes sont plus faciles que pour les produits, mais dans le même esprit. Nous ne donnons donc que la preuve pour le produit C = AB de deux séries différentiellement finies A et B. Par la formule de Leibniz, toutes les dérivées de

C s'écrivent comme combinaisons linéaires de produits entre une dérivée $A^{(i)}$ de A et une dérivée $B^{(j)}$ de B. Les dérivées de A et de B étant engendrées par un nombre fini d'entre elles, il en va de même pour les produits $A^{(i)}B^{(j)}$, ce qui prouve la finitude différentielle de C. ■

Exercice 14.4 Prouver le cas du produit de suites polynomialement récurrentes. ■

Cette preuve donne également un algorithme pour trouver l'équation différentielle (resp. la récurrence) cherchée : il suffit de calculer les dérivées (resp. les décalées) successives en les récrivant sur un ensemble fini de générateurs. Une fois en nombre suffisant (c'est-à-dire au pire égal à la dimension plus 1), elles satisfont une relation linéaire. À partir de la matrice dont les lignes contiennent les coordonnées des dérivées successives (resp. des décalés successifs) sur cet ensemble fini de générateurs, cette relation s'obtient par un calcul du noyau de la transposée. Le calcul se ramène donc ultimement à la résolution d'un système linéaire à coefficients dans $\mathbb{K}[X]$, ce qui peut se faire efficacement grâce à l'algorithme de Storjohann (Chapitre 11).

Exemple 14.8 Voici comment prouver (et même découvrir) l'identité

$$\arcsin(X)^2 = \sum_{k\geq 0} \frac{k!}{\left(\frac{1}{2}\right)\cdots\left(k+\frac{1}{2}\right)} \frac{X^{2k+2}}{2k+2}.$$

Le calcul consiste à partir d'une équation satisfaite par arcsin(X), en déduire une équation satisfaite par son carré, traduire cette équation en récurrence sur les coefficients, et conclure en constatant que cette récurrence est satisfaite par les coefficients de la série.

Le point de départ est la propriété

$$(\arcsin(X))' = \frac{1}{\sqrt{1-X^2}},$$

qui permet de représenter arcsin par l'équation différentielle $(1-X^2)y'' - Xy' = 0$ avec les conditions initiales $y(0) = 0$, $y'(0) = 1$.

Ensuite, en posant $h = y^2$, les dérivations et réductions successives donnent

$$\begin{aligned}
h' &= 2yy', \\
h'' &= 2y'^2 + 2yy'' = 2y'^2 + \frac{2X}{1-X^2}yy', \\
h''' &= 4y'y'' + \frac{2X}{1-X^2}(y'^2 + yy'') + \left(\frac{2}{1-X^2} + \frac{4X^2}{(1-X^2)^2}\right)yy', \\
&= \left(\frac{4X+2}{1-X^2} + \frac{6X^2}{(1-X^2)^2}\right)yy' + \frac{2X}{1-X^2}y'^2.
\end{aligned}$$

Les quatre vecteurs h, h', h'', h''' sont combinaisons linéaires des trois vecteurs y^2, yy', y'^2. Ils sont donc liés et une relation de liaison s'obtient en calculant le noyau

de la matrice 3×4 qui découle de ce système. Le résultat est

$$(1 - X^2)h''' - 3Xh'' - h' = 0.$$

La récurrence qui s'en déduit est

$$(n+1)(n+2)(n+3)a_{n+3} - (n+1)^3 a_{n+1} = 0.$$

Comme le facteur $(n+1)$ ne s'annule pas sur $\mathbb{N}$, il est possible de simplifier pour obtenir la récurrence équivalente

$$(n+2)(n+3)a_{n+3} - (n+1)^2 a_{n+1} = 0.$$

Si le membre droit de l'identité n'était pas connu, il serait possible de le déterminer grâce à cette récurrence à deux termes et aux conditions initiales $h(0) = 0$, $h'(0) = 0$, $h''(0) = 2$. Lorsque le membre droit est donné, la vérification que les coefficients de la série ci-dessus vérifient cette identité est facile.

Bornes

La preuve du Théorème 14.5 permet aussi de borner l'ordre des équations : l'ordre de l'équation satisfaite par une somme est borné par la somme des ordres des équations des sommants, et l'ordre de l'équation satisfaite par un produit est borné par le produit des ordres. On en déduit alors un algorithme simple de preuve d'identités.

Exemple 14.9 L'identité de Cassini sur les nombres de Fibonacci s'écrit

$$F_{n+2}F_n - F_{n+1}^2 = (-1)^n.$$

Pour prouver cette égalité, le point de départ est simplement la récurrence définissant les nombres de Fibonacci :

$$F_{n+2} = F_{n+1} + F_n,$$

qui exprime que tous les décalés de F_n sont des combinaisons linéaires de F_n et F_{n+1}. Les produits qui interviennent dans l'identité de Cassini s'expriment donc *a priori* comme combinaison linéaire de F_n^2, F_nF_{n+1} et F_{n+1}^2 et donc le membre de gauche vérifie une récurrence d'ordre borné par 4. Le membre droit vérifiant une récurrence d'ordre 1, leur différence vérifie une récurrence d'ordre au plus 5. *Il n'est pas nécessaire de calculer cette récurrence*. Il suffit de vérifier que ses 5 conditions initiales sont nulles. Autrement dit, vérifier l'identité pour $n = 0, \ldots, 4$ la prouve !

Si le membre droit n'est pas donné, on calcule la récurrence satisfaite par le membre gauche en récrivant ses décalés sur les générateurs $F_{n+1}F_n, F_n^2, F_{n+1}^2$:

$$\begin{aligned} u_n &= F_{n+2}F_n - F_{n+1}^2 = F_{n+1}F_n + F_n^2 - F_{n+1}^2, \\ u_{n+1} &= F_{n+2}F_{n+1} + F_{n+1}^2 - F_{n+2}^2 = F_{n+1}^2 - F_n^2 - F_nF_{n+1} = -u_n. \end{aligned}$$

et on conclut en observant que $u_0 = 1$.

Exercice 14.5 De la même manière, montrer l'identité $\sin^2 X + \cos^2 X = 1$, avec et sans calcul. ■

Produit d'Hadamard

Corollaire 14.8 Si $A = \sum_{n\geq 0} a_n X^n$ et $B = \sum_{n\geq 0} b_n X^n$ sont deux séries différentiellement finies, alors leur produit d'Hadamard $A \odot B = \sum_{n\geq 0} a_n b_n X^n$ l'est aussi.

La preuve est également un algorithme : des deux équations différentielles se déduisent deux récurrences satisfaites par les suites (a_n) et (b_n) ; d'après la section précédente, le produit $(a_n b_n)$ vérifie alors une récurrence linéaire, dont se déduit enfin l'équation différentielle satisfaite par sa série génératrice.

Exemple 14.10 Les polynômes orthogonaux de Hermite ont pour série génératrice

$$\sum_{n\geq 0} H_n(X)\frac{Z^n}{n!} = \exp(Z(2X - Z)).$$

À partir de là, la détermination du membre droit de l'identité suivante due à Mehler est entièrement algorithmique :

$$\sum_{n\geq 0} H_n(X)H_n(Y)\frac{Z^n}{n!} = \frac{\exp\left(\frac{4Z(XY - Z(X^2+Y^2))}{1-4Z^2}\right)}{\sqrt{1-4Z^2}}.$$

Exemple 14.11 Si l'intégrale

$$I(t) = \int_{-\infty}^{\infty} e^{-xt^2} f(x)\,dx,$$

peut être calculée à partir du développement de Taylor de f en intervertissant sommation et intégration, alors $I(t)$ est différentiellement finie si f l'est. Il en va de même pour la transformée de Laplace ou la transformée de Fourier.

Exercice 14.6 Montrer que dans les mêmes conditions, f est différentiellement finie si I l'est, et donner un algorithme pour calculer une équation différentielle linéaire satisfaite par f à partir d'une équation satisfaite par I. ■

14.3 Séries algébriques

Définition 14.3 Une série formelle $F(X) \in \mathbb{K}[[X]]$ est une *série algébrique* lorsqu'il existe un polynôme non nul $P(X, Y) \in \mathbb{K}[X, Y]$ tel que $P(X, F(X)) = 0$.

Théorème 14.9 Si la série $F(X)$ annule un polynôme $P(X,Y) \in \mathbb{K}[X,Y]$ de degré d en Y, où $\mathbb{K}$ est un corps de caractéristique 0, alors elle est solution d'une équation différentielle linéaire d'ordre au plus d.

Démonstration. La preuve est algorithmique. Quitte à diviser d'abord P par son pgcd avec sa dérivée P_Y par rapport à Y, il est possible de le supposer premier avec P_Y (car la caractéristique est nulle !). En dérivant $P(X,Y) = 0$ et en isolant Y', il vient

$$Y' = -\frac{P_X}{P_Y}.$$

Par *inversion modulaire* de P_Y (voir le Chapitre 6), cette identité se récrit via un calcul de pgcd étendu en

$$Y' = R_1(Y) \bmod P,$$

où R_1 est un polynôme en Y de degré au plus d et à coefficients dans $\mathbb{K}(X)$. Ceci signifie que Y' s'écrit comme combinaison linéaire de $1, Y, Y^2, \ldots, Y^{d-1}$ à coefficients dans $\mathbb{K}(X)$. Dériver à nouveau cette équation, puis récrire Y' et prendre le reste de la division par P mène à nouveau à une telle combinaison linéaire pour Y'' et plus généralement pour les dérivées successives de Y. Les $d+1$ vecteurs $Y, Y', \ldots, Y^{(d)}$ sont donc linéairement dépendants et la relation de liaison est l'équation cherchée. ■

Exemple 14.12 Les dénombrements d'arbres mènent naturellement à des équations algébriques sur les séries génératrices. Ainsi, la série génératrice $C(X)$ des nombres de Catalan (nombre d'arbres binaires à n sommets internes) vérifie

$$C = 1 + XC^2;$$

la série génératrice $M(X)$ des nombres de Motzkin (le nombre d'arbres unaires-binaires à n sommets internes) vérifie

$$M = 1 + XM + XM^2.$$

Dans les deux cas, il est aisé d'obtenir d'abord une équation différentielle puis une récurrence qui permet de calculer efficacement ces nombres par scindage binaire (Chapitre 15). Dans le cas des nombres de Catalan, la récurrence est d'ordre 1, la suite est donc hypergéométrique et s'exprime aisément.

Exercice 14.7 Trouver une formule explicite des nombres de Catalan. Récrire les fonctions Γ qui pourraient apparaître dans le résultat en termes de factorielles, puis de coefficient binomial. ■

Exercice 14.8 À l'aide d'un système de calcul formel, calculer une récurrence linéaire satisfaite par les coefficients de la série $Y(X)$ solution de $Y = 1 + XY + XY^7$. ■

Les mêmes arguments que ci-dessus mènent à une autre propriété de clôture des séries différentiellement finies.

Corollaire 14.10 Si F est une série différentiellement finie et A une série algébrique sans terme constant, alors $F \circ A$ est différentiellement finie.

La preuve consiste à observer que les dérivées successives de $F \circ A$ s'expriment comme combinaisons linéaires des $F^{(i)}(A)A^j$ pour un nombre fini de dérivées de F (par finitude différentielle) et de puissances de A (par la même preuve que pour le Théorème 14.9). Cette preuve fournit encore un algorithme.

Exemple 14.13 À l'aide d'un système de calcul formel, calculer une récurrence linéaire satisfaite par les coefficients du développement en série de Taylor de

$$\exp\left(\frac{1-\sqrt{1-4X}}{2}\right).$$

Corollaire 14.11 Si A est une série algébrique, alors $\exp \int A$ est différentiellement finie.

Exercice 14.9 Faire la preuve. ∎

Exemple 14.14 — La moyenne arithmético-géométrique. Si a et b sont deux réels tels que $0 \le b \le a$, les deux suites définies par

$$a_{n+1} = \frac{a_n + b_n}{2}, \quad b_{n+1} = \sqrt{a_n b_n}, \quad a_0 = a, \quad b_0 = b$$

ont une limite commune dont l'existence se déduit de $b_n \le b_{n+1} \le a_{n+1} \le a_n$ et

$$a_{n+1}^2 - b_{n+1}^2 = \left(\frac{a_n - b_n}{2}\right)^2.$$

Cette limite est notée $M(a,b)$. L'identité suivante

$$M(a,b) = \frac{a}{F(\frac{1}{2},\frac{1}{2};1;1-\frac{b^2}{a^2})}.$$

due à Gauss, où la série hypergéométrique est définie comme

$$F(a,b;c;z) := \sum_{n\ge 0} \frac{(a)_n (b)_n}{(c)_n} \frac{z^n}{n!}, \quad \text{où} \quad (x)_n = x(x+1)\cdots(x+n-1),$$

peut se prouver par les algorithmes de ce chapitre.

Tout d'abord, la fonction M est clairement homogène : pour $\lambda > 0$, $M(\lambda a, \lambda b) = \lambda M(a,b)$ ce qui permet de concentrer l'étude sur la fonction de une variable $M(1,x)$,

dont on peut prouver qu'elle est analytique au voisinage de $x = 1$. Ensuite, l'homogénéité et le fait que $\mathrm{M}(a_1, b_1) = \mathrm{M}(a_0, b_0)$ avec $a_0 = 1 + x$ et $b_0 = 1 - x$ fournissent l'identité

$$\mathrm{M}(1, \sqrt{1 - x^2}) = (1 + x)\mathrm{M}\left(1, \frac{1 - x}{1 + x}\right).$$

En posant $\mathrm{M}(1, x) = 1/\mathrm{A}(1 - x^2)$, cette équation entraine l'équation fonctionnelle

$$(1 + x)\mathrm{A}(x^2) - \mathrm{A}\left(1 - \frac{(1 - x)^2}{(1 + x)^2}\right) = 0, \tag{14.4}$$

qui admet une seule solution série avec $\mathrm{A}(0) = 1$. Il suffit donc de montrer que cette solution est $f(x) = \mathrm{F}(\frac{1}{2}, \frac{1}{2}; 1; x)$.

Pour cela, on va construire une équation différentielle satisfaite par le membre gauche de l'équation ci-dessus évalué en $\mathrm{A}(x) = f(x)$, puis observer que quelques conditions initiales suffisent à prouver la nullité. La définition de la série hypergéométrique entraîne clairement une récurrence d'ordre 1 sur les coefficients

$$\frac{u_{n+1}}{u_n} = \frac{(a + n)(b + n)}{(c + n)(1 + n)}.$$

En convertissant cette récurrence en équation différentielle, on obtient que $f(x)$ satisfait

$$4x(x - 1)f''(x) + 4(x - 2)f'(x) + f(x) = 0.$$

À partir de cette équation, on construit une équation pour $g(x) = f(x^2)$:

$$x(x^2 - 1)g''(x) + (3x^2 - 1)g'(x) + xg(x) = 0$$

et une autre pour $h(x) = g(1 - (1 - x)^2/(1 + x)^2)$:

$$x(x - 1)(x + 1)^2 h''(x) + (x + 1)(x^2 + 2x - 1)h'(x) - (x - 1)h(x) = 0 \tag{14.5}$$

(il s'agit de cas simples du corollaire ci-dessus). En utilisant la clôture par somme, on peut alors construire une équation pour $(1 + x)g(x) + h(x)$, et on regrouve exactement la même que pour h (ceci peut se voir directement : en remplaçant $g(x)$ dans l'équation qui le définit par $u(x)/(1 + x)$, on obtient l'équation satisfaite par h).

À ce stade, l'équation (14.5) est donc satisfaite par le membre gauche de l'équation (14.4) lorsque A vaut f et il s'agit de prouver que cette solution de (14.5) est nulle. Le Corollaire 14.3 ne s'applique pas ici puisque le coefficient de tête s'annule en 0. En revanche, on peut passer à la récurrence satisfaite par les coefficients des solutions séries de cette équation (14.5) :

$$(n + 3)^2 u_{n+3} + (n^2 + 2n - 1)u_{n+2} - (n^2 + 4n + 2)u_{n+1} - n^2 u_n = 0,$$

ce qui montre qu'il suffit de vérifier les conditions initiales $h(0) = h'(0) = h''(0) = 0$. Pour cela, aucun calcul n'est nécessaire puisque la série A est définie comme solution de (14.4), ce qui conclut.

Dans cet exemple, nous sommes partis de l'équation satisfaite par la série hypergéométrique pour prouver qu'elle vérifiait une équation fonctionnelle. Il est également possible d'utiliser une méthode de coefficients indéterminés pour calculer suffisamment de coefficients de la solution $A(x)$ de (14.4), puis conjecturer cette équation différentielle par un calcul d'approximants de Padé–Hermite (Chapitre 7), et enfin de montrer comme nous l'avons fait que la série définie par cette équation conjecturée est bien solution de l'équation fonctionnelle.

14.4 Au-delà

En général, la composition de deux séries différentiellement finies n'est pas différentiellement finie. Voici trois résultats plus forts, dont la preuve repose sur la théorie de Galois différentielle et dépasse le cadre de cet ouvrage.

Théorème 14.12 Soient A et F deux séries de $\mathbb{K}[[X]]$.
1. Les séries F et 1/F sont simultanément différentiellement finies si et seulement si F'/F est algébrique.
2. Les séries F et $\exp(\int F)$ sont simultanément différentiellement finies si et seulement si F est algébrique.
3. Soit A algébrique de genre supérieur ou égal à 1, alors F et $A \circ F$ sont différentiellement finies si et seulement si F est algébrique.

Exercice 14.10 Prouver le sens « si » de ces trois propriétés. ■

Exercices

Exercice 14.11 Parmi les suites données par les termes généraux suivants, indiquer celles qui sont polynomialement récurrentes et celles qui ne le sont pas :

$$\frac{1}{\binom{2n}{n}}, \qquad \cos n, \qquad 2^{2^n}, \qquad H_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n}, \qquad \lfloor \log n \rfloor.$$

Parmi les développements en série des fonctions suivantes, indiquer ceux qui sont différentiellement finis et ceux qui ne le sont pas :

$$\exp\left(\sqrt[3]{1-X}\right), \qquad \prod_{i\geq 1} \frac{1}{1-X^i}, \qquad \sum_{i\geq 0} X^{2^i}, \qquad \sum_{n\geq 0} F_n^3 \frac{X^n}{n!}, \qquad \tan(X).$$

(F_n désigne le n-ième nombre de Fibonacci, dont la suite vérifie $F_{n+2} = F_{n+1} + F_n$ et $F_1 = F_0 = 1$.) ■

Exercice 14.12
1. Soit $(a_n)_{n\geq 0}$ une suite polynomialement récurrente d'éléments d'un corps de caractéristique 0. Montrer que les suites extraites $(a_{2n})_{n\geq 0}$ et $(a_{2n+1})_{n\geq 0}$ sont

également polynomialement récurrentes.

2. Soit $(g_n)_{n\geq 0}$ une suite dont la série génératrice exponentielle est

$$\sum_{n\geq 0} g_n \frac{X^n}{n!} = \frac{\exp(-X/2 - X^2/4)}{\sqrt{1-X}}.$$

Montrer que la suite $(g_n)_{n\geq 0}$ satisfait une récurrence linéaire homogène, d'ordre au plus 3 et à coefficients polynomiaux de degrés au plus 2. La récurrence n'est pas demandée explicitement.

3. Si $N > 0$, quel est le coût binaire, en fonction de N, du calcul de g_N ? [Utiliser des résultats du Chapitre 15.]

■

Exercice 14.13 — Opérations de clôture pour les équations différentielles linéaires à coefficients constants. Soient $f(X)$ et $g(X)$ solutions des équations différentielles linéaires homogènes

$$a_m f^{(m)}(X) + \cdots + a_0 f(X) = 0, \qquad b_n g^{(n)}(X) + \cdots + b_0 g(X) = 0,$$

avec $a_0, \ldots, a_m, b_0, \ldots, b_n$ des coefficients rationnels.

1. Montrer que $f(X)g(X)$ et $f(X) + g(X)$ sont solutions d'équations différentielles du même type.

Si le polynôme $a_m X^m + \cdots + a_0$ se factorise sur $\mathbb{C}$ en

$$a_m (X - \alpha_1)^{d_1} \cdots (X - \alpha_k)^{d_k},$$

on rappelle qu'une base de l'espace des solutions de

$$a_m f^{(m)}(X) + \cdots + a_0 f(X) = 0$$

est donnée par $\left\{e^{\alpha_1 X}, Xe^{\alpha_1 X}, \ldots, X^{d_1-1}e^{\alpha_1 X}, \ldots,\ e^{\alpha_k X}, Xe^{\alpha_k X}, \ldots, X^{d_k-1}e^{\alpha_k X}\right\}$.

2. Montrer qu'une équation satisfaite par $f(X) + g(X)$ peut être calculée à l'aide de l'algorithme d'Euclide.
3. Montrer qu'une équation satisfaite par $f(X)g(X)$ peut être obtenue par un calcul de résultant.

■

Exercice 14.14 — Puissance symétrique d'équation différentielle. Soient m et d deux entiers naturels et soient $a(X), b(X)$ deux polynômes dans $\mathbb{Q}[X]$ de degrés au plus d.

1. Montrer qu'il existe une équation différentielle, notée $\mathcal{E}_m$, linéaire homogène d'ordre $m+1$, à coefficients dans $\mathbb{Q}[X]$, qui admet $\phi(X)^m$ comme solution, quelle que soit la solution $\phi(X)$ de l'équation différentielle

$$(\mathcal{E}_1) \qquad y''(X) + a(X)y'(X) + b(X)y(X) = 0.$$

On admettra que pour toute base (ϕ_1, ϕ_2) de solutions de $\mathcal{E}_1$, les fonctions $\phi_1^i \phi_2^{m-i}$, $i = 0, \ldots, m$ sont linéairement indépendantes.

2. Montrer que si $a = 0$, alors $\mathcal{E}_2 : y'''(\mathrm{X}) + 4b(\mathrm{X})y'(\mathrm{X}) + 2b'(\mathrm{X})y(\mathrm{X}) = 0$.
3. Expliciter un algorithme pour calculer $\mathcal{E}_m$, en ramenant le calcul final à un calcul sur des matrices de polynômes.
4. Estimer la complexité de cet algorithme en nombres d'opérations dans $\mathbb{Q}$ en fonction de m et d.
5. Pour m fixé, on associe à une fonction y de X des fonctions L_k par les valeurs initiales
$$\mathrm{L}_0(\mathrm{X}) = y(\mathrm{X}), \quad \mathrm{L}_1(\mathrm{X}) = y'(\mathrm{X})$$
et la relation de récurrence
$$\mathrm{L}_{k+1}(\mathrm{X}) = \mathrm{L}_k'(\mathrm{X}) + ka(\mathrm{X})\mathrm{L}_k(\mathrm{X}) + k(m-k+1)b(\mathrm{X})\mathrm{L}_{k-1}(\mathrm{X}).$$
   a. Montrer que lorsque $y = \phi^m$, pour $0 \le k \le m$,
   $$\mathrm{L}_k(\mathrm{X}) = m(m-1)\ldots(m-k+1)\phi^{m-k}(\mathrm{X})\phi'(\mathrm{X})^k,$$
   et $\mathrm{L}_{m+1}(\mathrm{X}) = 0$.
   b. Montrer que $\mathrm{L}_{m+1}(\mathrm{X}) = 0$ n'est autre que l'équation $\mathcal{E}_m$.
   c. En déduire des bornes sur les degrés des coefficients de $\mathcal{E}_m$ et un algorithme de complexité $\mathrm{O}(m^3\mathsf{M}(d))$ pour le calcul de $\mathcal{E}_m$.
6. Montrer que si $a, b \in \mathbb{Q}$, alors $\mathcal{E}_m$ est à coefficients constants et qu'on peut calculer $\mathcal{E}_m$ en $\mathrm{O}(\mathsf{M}(m)\log m)$ opérations dans $\mathbb{Q}$.

■

Notes

La première édition du *Handbook of Mathematical Functions* [AS92] remonte à 1964 et a été suivie de huit rééditions ; le *NIST Handbook of Mathematical Functions* [Olv+10] est vu comme un successeur de l'œuvre d'origine. Le site web [Slo06] est le prolongement en ligne du livre éponyme *Encyclopedia of Integer Sequences*[SP95] et est interrogeable directement depuis le système Sage.

Les propriétés de clôture des séries différentiellement finies ont été décrites avec leurs applications par Stanley [Sta80 ; Sta99] et Lipshitz [Lip89].

L'utilisation des séries différentiellement finies pour les séries algébriques en combinatoire est exploitée à de nombreuses reprises par Comtet dans son livre [Com70], où il utilise l'algorithme décrit dans la preuve du Théorème 14.9. L'histoire de cet algorithme est compliquée. Il était connu d'Abel qui l'avait rédigé dans un manuscrit de 1827 qui n'a pas été publié. Ce manuscrit est décrit (page 287) dans les œuvres complètes d'Abel [Abe92]. Ensuite, ce résultat a été retrouvé par Sir James Cockle en 1860 et popularisé par le révérend Harley en 1862 [Har62]. Quelques années plus tard, il est encore retrouvé par Tannery dans sa thèse, dont le manuscrit est remarquablement clair [Tan74]. Pour le calcul du développement de séries algébriques, il faut tenir compte de l'ordre assez élevé des équations obtenues [Bos+07b], ce qui rend cet algorithme utile pour des très grandes précisions, l'itération de Newton du Chapitre 3 étant préférable pour des précisions modérées.

Les limitations de la dernière section ne sont pas très connues. Elles sont dues à Harris et Sibuya pour la première [HS85] et à Singer pour les deux autres [Sin86].

En ce qui concerne les algorithmes et les implantations, la plupart des algorithmes présentés dans ce chapitre sont implantés dans le package `gfun` de Maple [SZ94]. L'algorithme rapide de la Section 14.1 provient essentiellement de la thèse de doctorat de Bostan [Bos03], qui donne une variante légèrement plus efficace (d'un facteur constant).

Bibliographie

Abe92 Abel, Niels Henrik (1992). *Œuvres complètes. Tome II*. Réimpression de la seconde édition (1881). Éditions Jacques Gabay. url : `http://www.gallica.fr`.

AS92 Abramowitz, Milton et Irene A. Stegun, éds. (1992). *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Réimpression de l'édition de 1972. Dover Publications Inc.

Bos+07b Bostan, Alin, Frédéric Chyzak, Grégoire Lecerf, Bruno Salvy et Éric Schost (2007). « Differential equations for algebraic functions ». In : *ISSAC'07 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Dongming Wang. ACM Press, p. 25–32.

Bos03 Bostan, Alin (2003). « Algorithmique efficace pour des opérations de base en calcul formel ». Thèse de Doctorat. École polytechnique.

Com70 Comtet, L. (1970). *Analyse combinatoire*. 2 volumes. Presses Universitaires de France.

Har62 Harley, Rev. Robert (1862). « On the theory of the transcendental solution of algebraic equations ». In : *Quarterly Journal of Pure and Applied Mathematics*, vol. 5, p. 337–360.

HS85 Harris, William A. et Yasutaka Sibuya (1985). « The reciprocals of solutions of linear ordinary differential equations ». In : *Advances in Mathematics*, vol. 58, n°2, p. 119–132.

Lip89 Lipshitz, L. (1989). « D-finite power series ». In : *Journal of Algebra*, vol. 122, n°2, p. 353–373.

Olv+10 Olver, Frank W. J., Daniel W. Lozier, Ronald F. Boisvert et Charles W. Clark, éds. (2010). *NIST handbook of mathematical functions*. U.S. Department of Commerce, National Institute of Standards et Technology, Washington, DC ; Cambridge University Press, Cambridge.

Sin86 Singer, Michael F. (1986). « Algebraic relations among solutions of linear differential equations ». In : *Transactions of the American Mathematical Society*, vol. 295, n°2, p. 753–763.

Slo06 Sloane, N. J. A. (2006). *The on-line encyclopedia of integer sequences*. url : `http://www.research.att.com/~njas/sequences/`.

SP95 Sloane, N. J. A. et Simon Plouffe (1995). *The encyclopedia of integer sequences*. Academic Press, Inc., San Diego, CA.

Sta80 Stanley, R. P. (1980). « Differentiably finite power series ». In : *European Journal of Combinatorics*, vol. 1, n°2, p. 175–188.

Sta99 Stanley, Richard P. (1999). *Enumerative combinatorics*. Vol. 2. Cambridge University Press.

SZ94 Salvy, Bruno et Paul Zimmermann (1994). « Gfun : a Maple package for the manipulation of generating and holonomic functions in one variable ». In : *ACM Transactions on Mathematical Software*, vol. 20, n°2, p. 163–177.

Tan74 Tannery, Jules (1874). « Propriétés des intégrales des équations différentielles linéaires à coefficients variables ». Thèse de doctorat ès sciences mathématiques. Faculté des Sciences de Paris. url : http://gallica.bnf.fr.

15. Récurrences linéaires à coefficients polynomiaux : N^e terme, N premiers termes

Résumé

Pour calculer les N premiers termes d'une suite polynomialement récurrente, l'algorithme direct par déroulement de la récurrence est quasi-optimal vis-à-vis de N : sa complexité arithmétique (resp. binaire) est linéaire en N (resp. quasi-quadratique en N). Le N-ième terme d'une telle suite peut être calculé plus rapidement que l'ensemble des N premiers termes, en essentiellement $\sqrt{N}$ opérations arithmétiques et N opérations binaires. Ces calculs rapides sont la base de l'évaluation numérique à grande précision en complexité quasi-optimale pour les solutions d'équations différentielles linéaires.

Le Chapitre 4 a montré comment calculer le N-ième terme d'une suite donnée par une récurrence à coefficients constants en complexité arithmétique $O(\log N)$. Ce chapitre attaque le cadre plus général des récurrences à coefficients polynomiaux.

Le calcul du N-ième terme ou des N premiers termes d'une suite polynomialement récurrente, donnée par la récurrence à coefficients polynomiaux

$$p_r(n)u_{n+r} + \cdots + p_0(n)u_n = 0, \qquad n \in \mathbb{N}, \tag{15.1}$$

intervient fréquemment en combinatoire, ou pour calculer des troncatures de séries, ainsi que comme étape clé dans des algorithmes de recherche de solutions polynomiales d'équations fonctionnelles linéaires, dans un algorithme de factorisation déterministe d'entiers, ou encore en cryptologie, dans des questions de comptage de points sur des courbes. Ce calcul intervient aussi dans l'évaluation numérique efficace des séries différentiellement finies au Chapitre 17.

Définition 15.1 La récurrence (15.1) est dite *régulière sur l'intervalle* $\{0,\dots,N\}$ lorsque le coefficient de tête p_r de la récurrence (15.1) ne s'annule sur aucun des entiers $0,\dots,N-r$. Elle est dite *régulière* si cette propriété est vraie pour tout N, et *singulière* sinon.

En dehors de sa dernière Section 15.4, ce chapitre traite de récurrences régulières.

Si les coefficients $p_i(n)$ de la récurrence (15.1) ont degré au plus d en n, l'algorithme naïf par déroulement de la récurrence a une complexité arithmétique en $O(rdN)$ et une complexité binaire en $O\big(rN^2 M_{\mathbb{Z}}(d\log N)\big)$ (voir §15.1). Ces complexités sont quasi-optimales vis-à-vis de N pour le calcul de l'ensemble des N premiers termes.

Pour le calcul du N-ième terme seul, les algorithmes présentés dans ce chapitre n'ont pas une aussi bonne complexité que pour des coefficients constants, mais cette fois encore, ils sont plus efficaces que le simple déroulement de la récurrence. Un nouveau phénomène distingue ces algorithmes de ceux qui ont été présentés jusqu'ici : les idées qui permettent le calcul du N-ième terme en bonne complexité arithmétique ne sont pas les mêmes que pour abaisser la complexité binaire. Ainsi, l'algorithme par « *pas de bébés, pas de géants* » de la Section 15.2 donne un gain en racine carrée de N sur la complexité arithmétique, par rapport à la méthode naïve, mais ne gagne rien sur la complexité binaire. Quant à l'algorithme par *scindage binaire* de la Section 15.3, il n'abaisse en rien la complexité arithmétique, et améliore pourtant la complexité binaire jusqu'à la rendre quasi-linéaire en la taille de sa sortie.

15.1 Calcul naïf de N! et de suites polynomialement récurrentes

Cas de la factorielle

La définition de la factorielle comme produit,

$$N! = 1 \times 2 \times \cdots \times N,$$

montre que N! peut être calculé en $N-1$ opérations arithmétiques, ce qui est optimal pour le calcul des N premières factorielles.

L'estimation de la complexité binaire de cette méthode repose sur la *formule de Stirling* :

$$\log N! = N\log N - N + \frac{1}{2}\log N + O(1), \qquad N \to \infty. \tag{15.2}$$

En particulier, l'équivalence $\log N! \sim N\log N$ donne la taille de N!. La k-ième étape du produit multiplie $k!$ par $k+1$, soit donc un entier de taille $\log k! \sim k\log k$ avec un entier de taille $\log(k+1) \sim \log k$. Ce produit déséquilibré coûte donc moins de $ckM_{\mathbb{Z}}(\log k)$ opérations binaires pour une certaine constante c. Le calcul complet de N! par la méthode naïve effectue donc moins de

$$\sum_{k=1}^{N} ckM_{\mathbb{Z}}(\log k) = O\big(N^2 M_{\mathbb{Z}}(\log N)\big)$$

opérations binaires. La formule de Stirling montre que cette complexité binaire est quasi-optimale pour le calcul conjoint des nombres 1!, 2!, …, N!.

Cas général — complexité arithmétique

Pour une suite polynomialement récurrente définie l'équation (15.1), le calcul par l'algorithme direct de u_{n+r} à partir des r valeurs précédentes demande $O(rd)$ opérations arithmétiques pour l'évaluation des polynômes (par exemple par le schéma de Horner) puis $O(r)$ opérations pour effectuer la combinaison linéaire des u_{n+i}. Le calcul naïf de $u_0,\dots,u_N$ à partir des valeurs initiales u_0, …, u_{r-1} demande donc $O(rdN)$ opérations arithmétiques.

Une amélioration de cette borne vient de l'observation que les coefficients $p_i(n)$ peuvent être évalués sur les entiers $n = 0,1,\dots,N$ en utilisant des techniques d'évaluation multipoint rapide, avant de procéder au déroulement de (15.1). Cela permet d'abaisser la complexité arithmétique du calcul de $O(rdN)$ à

$$O(r\max(N,d)M(d)\log d/d),$$

voire à $O(r\max(N,d)M(d)/d)$ en utilisant l'observation du Corollaire 4.12 page 89, ce qui revient dans les deux cas à $\tilde{O}(r\max(N,d))$ si la multiplication polynomiale est à base de FFT.

Cas général — complexité binaire

Lorsque les coefficients des polynômes p_i sont entiers, la complexité binaire de l'algorithme naïf découle de nouveau essentiellement de la croissance de u_N. Pour estimer celle-ci, il est commode d'adopter une vision matricielle et de faire apparaître le rationnel u_N comme un quotient d'entiers. La suite vectorielle des $U_n = {}^t(u_n,\dots,u_{n+r-1})$ vérifie la récurrence du premier ordre

$$U_{n+1} = \frac{1}{p_r(n)}A(n)U_n \quad \text{avec} \quad A(n) = \begin{pmatrix} 0 & p_r(n) & & \\ & 0 & p_r(n) & \\ & & \ddots & \ddots \\ & & & 0 & p_r(n) \\ -p_0(n) & & \dots & & -p_{r-1}(n) \end{pmatrix}.$$

Avec la suite hypergéométrique (w_n) définie par la condition initiale $w_0 = 1$ et la récurrence

$$w_{n+1} = p_r(n)w_n, \qquad (n \in \mathbb{N}),$$

qui permet de récrire U_n sous la forme

$$U_n = \frac{1}{w_n}\Big(A(n-1)\cdots A(0)\Big)U_0,$$

le rationnel u_N s'écrit $u_N = v_N/w_N$, où v_N est le premier coefficient du vecteur $\Big(A(N-1)\cdots A(0)\Big)U_0$.

La croissance de v_N est estimée à l'aide de la norme d'indice 1 et de sa norme subordonnée. Pour un vecteur U de dimension r et une matrice M de dimension $r \times r$, il s'agit de

$$\|U\| = \sum_{j=1}^{r}|v_j| \qquad \text{et} \qquad \|M\| = \max_{1\le j\le r}\sum_{i=1}^{r}|m_{i,j}|.$$

Il est alors facile de vérifier (et c'est le sens de la norme subordonnée) que pour tout u, $\|\mathrm{M}u\| \le \|\mathrm{M}\|\|u\|$. Soit ℓ un majorant commun pour la taille binaire des coefficients des p_i, $0 \le i \le r$, et de même ℓ' un majorant de la taille des conditions initiales u_j, $0 \le j < r$. Il s'ensuit que la norme de $\mathrm{A}(n)$ est bornée par

$$\|\mathrm{A}(n)\| \le r\,2^{\ell}(d+1)(n+1)^d,$$

et celle de U_0 par $r2^{\ell'}$. Les majorations

$$|v_{\mathrm{N}}| \le \|\mathrm{A}(\mathrm{N}-1)\|\cdots\|\mathrm{A}(0)\|\|\mathrm{U}_0\| \le r^{\mathrm{N}+1}\,2^{\ell\mathrm{N}+\ell'}(d+1)^{\mathrm{N}}(\mathrm{N}!)^d$$

fournissent la borne $\mathrm{O}\big(d\mathrm{N}\log\mathrm{N} + \mathrm{N}\ell + \ell' + \mathrm{N}\log r\big)$ sur la taille $\log|v_{\mathrm{N}}|$. Par le même raisonnement, la taille du dénominateur w_{N} est du même ordre (mêmes formules pour $r = 1$), si bien que par le même type d'argument que pour la factorielle, la complexité binaire du calcul naïf de u_{N} est $\mathrm{O}\big(r\mathrm{N}^2\mathrm{M}_{\mathbb{Z}}(d\log\mathrm{N} + \ell + \log r + \ell'/\mathrm{N})\big)$.

Conclusion

Les mêmes raisonnements sur la taille et sur la complexité s'appliquent sans modification au cas de récurrences inhomogènes

$$p_r(n)u_{n+r} + \cdots + p_0(n)u_n = q(n), \qquad n \in \mathbb{N}, \tag{15.3}$$

où q est lui aussi un polynôme. Nous avons donc obtenu le résultat suivant. L'algorithme correspondant dans le cas légèrement plus général des récurrences singulières est présenté en Algorithme 15.3 page 281.

Proposition 15.1 Étant donnée la récurrence (15.3) supposée régulière et des conditions initiales, le calcul des N premiers éléments de la suite peut être effectué en

$$\mathrm{O}\left(r\max(\mathrm{N}, d)\frac{\mathrm{M}(d)}{d}\right)$$

opérations arithmétiques, où d est une borne sur le degré des polynômes coefficients.

Si les coefficients de ces polynômes sont des entiers de taille bornée par ℓ et que les conditions initiales sont des rationnels de numérateurs et dénominateurs bornés par $\mathrm{O}(d\mathrm{N}\log\mathrm{N} + \ell\mathrm{N} + \mathrm{N}\log r)$, alors la complexité binaire de ce calcul est

$$\mathrm{O}\big(r\mathrm{N}^2\mathrm{M}_{\mathbb{Z}}(d\log\mathrm{N} + \ell + \log r)\big),$$

la taille du résultat étant $\mathrm{O}\big(\mathrm{N}^2(d\log\mathrm{N} + \ell + \log r)\big)$.

15.2 Pas de bébés, pas de géants

On a vu au Chapitre 4 que le N-ième terme d'une récurrence linéaire à coefficients constants peut être calculé en complexité arithmétique $\mathrm{O}(\log\mathrm{N})$. Dans le cas des

coefficients polynomiaux, les choses se compliquent : à ce jour, on ne connaît pas d'algorithme polynomial en $\log N$ (il est fort possible qu'un tel algorithme n'existe pas). L'exemple typique en est le calcul du N-ième terme de la factorielle $u_N = N!$, qui vérifie la récurrence à coefficients polynomiaux $u_{n+1} = (n+1)u_n$ pour $n \geq 0$. Une solution efficace exploite le théorème « évaluation interpolation » (Théorème 5.1). Elle utilise la technique des « pas de bébés, pas de géants » et requiert un nombre d'opérations arithmétiques en $\sqrt{N}$, à des facteurs logarithmiques près. Pour simplifier la présentation, supposons que N est un carré parfait. L'idée de l'algorithme est de poser

$$P(X) = (X+1)(X+2)\cdots(X+N^{1/2}),$$

afin d'obtenir la valeur de u_N à l'aide de l'équation

$$u_N = \prod_{j=0}^{N^{1/2}-1} P\left(jN^{1/2}\right). \tag{15.4}$$

Cette égalité suggère la procédure suivante :

1. *Pas de bébés.* Calculer les coefficients de P. Ceci requiert $O\left(M(\sqrt{N})\log N\right)$ opérations arithmétiques (en construisant un arbre binaire de feuilles $X+i$ comme en Section 5.4).
2. *Pas de géants.* Évaluer P sur les points $0, \sqrt{N}, 2\sqrt{N}, \ldots, (\sqrt{N}-1)\sqrt{N}$ et retrouver la valeur de u_N à l'aide de l'équation (15.4). En utilisant les techniques d'évaluation multipoint rapide (Section 5.4), ceci peut se faire également en $O\left(M(\sqrt{N})\log N\right)$ opérations arithmétiques.

Le coût total de cet algorithme est $O\left(M(\sqrt{N})\log N\right)$ opérations arithmétiques. Si la FFT est utilisée pour la multiplication des polynômes, le gain par rapport à la méthode directe est de l'ordre de $\sqrt{N}$, à des facteurs logarithmiques près, typique pour la technique des « pas de bébés, pas de géants ». La technique gagne encore par rapport à l'algorithme naïf si l'on emploie une multiplication par Karatsuba, mais ne présente aucun intérêt avec la multiplication naïve. Cette technique se généralise aux suites hypergéométriques quelconques ainsi qu'au calcul d'un terme d'une suite récurrente linéaire à coefficients polynomiaux, comme détaillé dans l'Algorithme 15.1.

Théorème 15.2 Étant donné la récurrence linéaire (15.1) des conditions initiales et un entier N, si la récurrence est régulière sur $\{0,\ldots,N\}$, alors l'Algorithme 15.1 est correct. Il calcule le N-ième terme d'une suite solution en

$$O\left(r^{\theta}M(\sqrt{dN})\log(dN)\right)$$

opérations arithmétiques, où d est une borne sur le degré des coefficients et θ est un exposant réalisable pour la complexité du produit de matrices (voir Chapitre 8).

Démonstration. L'algorithme calcule les produits $M = A(n-r)\cdots A(0)$ et $w = p_r(N-r)\cdots p_r(0)$ (pour ce dernier la récurrence est simplement $u_{n+1} = p_r(n)u_n$), puis le vecteur ${}^t(u_{N-r+1},\ldots,u_N)$ d'après la définition de A. La correction est donc claire.

Entrée Une récurrence de la forme (15.1) avec $\deg p_i \le d$, $i = 0,\dots,r$, des conditions initiales $u_0,\dots,u_{r-1}$, un entier N.
Sortie Les termes $u_{N-r+1},\dots,u_N$ de la solution.

1. Calculer $P(X) = A(X+m-1)\cdots A(X+1)A(X)$, pour $m = [(N/d)^{1/2}]$, où A est la matrice définie page 271.
2. Évaluer les coefficients des matrices $P(0), P(m), \dots, P(\lfloor \frac{N-r-m}{m} \rfloor m)$.
3. Évaluer ceux de $A(N-r), A(N-r-1), \dots, A(\lfloor \frac{N-r}{m} \rfloor m)$.
4. Calculer le produit de matrices scalaires $M = A(N-r)\cdots A(\lfloor \frac{N-r}{m} \rfloor m)P(\lfloor \frac{N-r}{m} \rfloor m)\cdots P(0)$.
5. Si $p_r(n) \neq 1$, calculer $w = p_r(N-r)\cdots p_r(0)$ par ce même algorithme ; sinon $w = 1$.
6. Renvoyer $\frac{1}{w} M \cdot {}^t(u_0,\dots,u_{r-1})$.

Algorithme 15.1 – Pas de bébés, pas de géants pour les récurrences linéaires.

Le produit de matrices de polynômes à l'étape (1) a degré au plus md. Il est effectué récursivement par un arbre des sous-produits comme au chapitre 5. Son coût est donc borné par $O(\mathsf{MM}(r,md)\log m)$ opérations arithmétiques. L'évaluation à l'étape (2) est effectuée en réalisant séparément les évaluations multipoints des r^2 coefficients de la matrice P par la technique du Corollaire 4.12 page 89 en $O(r^2 \frac{N}{m}\mathsf{M}(md)/(md))$ opérations arithmétiques. L'étape (3) effectue au plus m évaluations de matrices de degré au plus d, et est donc dominée par la précédente (qui calcule environ md évaluations de polynômes de degré md). L'étape (4) multiple $O(m + N/m)$ matrices scalaires en $O(\mathsf{MM}(r)(m + N/m))$ opérations. L'étape suivante a une complexité au plus égale à celle des précédentes. En résumé, la complexité arithmétique de cet algorithme est donc majorée par

$$\begin{aligned} &O\Big(\mathsf{MM}(r,md)\log m + r^2\frac{N}{m}\frac{\mathsf{M}(md)}{md} + \mathsf{MM}(r)\sqrt{Nd}\Big) \\ &\qquad = O\left(r^\theta \mathsf{M}(\sqrt{Nd})\log(Nd) + r^2\sqrt{Nd}\frac{\mathsf{M}(\sqrt{Nd})}{\sqrt{Nd}} + r^\theta\sqrt{Nd}\right), \\ &\qquad = O(r^\theta \mathsf{M}(\sqrt{Nd})\log(Nd)). \end{aligned}$$

■

Exercice 15.1 Étant donné un polynôme $f \in \mathbb{K}[X]$ de degré 2 et un entier positif N, borner le nombre d'opérations dans $\mathbb{K}$ suffisantes pour déterminer le coefficient de X^N du polynôme f^N. (Indication : La solution consiste à calculer par exponentiation binaire tous les coefficients $u_0, \dots, u_N$ de f^N mod X^{N+1}. Une approche encore plus rapide repose sur le fait que les u_n satisfont à une récurrence à coefficients polynomiaux d'ordre 2.) ■

Factorisation déterministe des entiers

Pour factoriser un entier N, tester tous les diviseurs plus petits que $\sqrt{N}$ a une complexité binaire au moins linéaire en $\sqrt{N}$. Afin d'accélérer ce calcul, on peut

rassembler tous les entiers plus petits que $\sqrt{N}$ en $\sqrt[4]{N}$ blocs, chacun contenant $\sqrt[4]{N}$ entiers consécutifs. Soit c de l'ordre de $\sqrt[4]{N}$ et $f_0 = 1\cdots c \bmod N$, $f_1 = (c+1)\cdots(2c) \bmod N$, …, $f_{c-1} = (c^2-c+1)\cdots(c^2) \bmod N$. Si les valeurs $f_0,\dots,f_{c-1}$ sont connues, alors il devient facile de déterminer un facteur de N, en prenant des pgcd de $f_0,\dots,f_{c-1}$ avec N : la complexité binaire associée est $O\big(\sqrt[4]{N}\,M_{\mathbb{Z}}(\log N)\log\log N\big)$.

Ainsi, la principale difficulté est de calculer les valeurs f_i. Pour ce faire, on prend $\mathbb{A} = \mathbb{Z}/N\mathbb{Z}$ et F le polynôme $(X+1)\cdots(X+c) \in \mathbb{A}[X]$, qu'on évalue en les points $0, c, 2c, \dots, c(c-1)$ en $O(M(c)\log c)$ opérations de $\mathbb{A}$. Par le théorème « évaluation interpolation », puisque c est d'ordre $\sqrt[4]{N}$, la complexité pour calculer les f_i est de $O(M(\sqrt[4]{N})\log N)$ opérations modulo N, soit $O(M(\sqrt[4]{N})\,M_{\mathbb{Z}}(\log N)\log N)$ opérations binaires. Ce terme est dominant et est donc aussi la complexité totale du calcul.

15.3 Scindage binaire

Cas de la factorielle

D'après la formule de Stirling (15.2), la factorielle de N/2 fait à peu près la moitié de la taille de N!. Il en va donc de même pour le produit des entiers de N/2 à N. En utilisant cette observation récursivement, l'idée de la méthode de scindage binaire consiste à exploiter la multiplication rapide en équilibrant les produits, calculant $P(a,b) = (a+1)(a+2)\cdots b$ récursivement par

$$P(a,b) = P(a,m)P(m,b) \quad \text{où} \quad m = \left\lfloor \frac{a+b}{2} \right\rfloor.$$

Soit $C(a,b)$ le coût binaire du calcul de $P(a,b)$. Il résulte immédiatement de la méthode que ce coût vérifie l'inégalité

$$C(a,b) \le C(a,m) + C(m,b) + M_{\mathbb{Z}}\Big(\max(\log P(a,m), \log P(m,b))\Big),$$

Sous l'hypothèse raisonnable que la complexité de la multiplication d'entiers est croissante avec la taille des entiers, le coût du calcul de $P(a,m)$ est inférieur au coût du calcul de $P(m,b)$, d'où la nouvelle inégalité

$$C(a,b) \le 2C(m,b) + M_{\mathbb{Z}}\Big(\log P(m,b)\Big).$$

À ce stade, le théorème « diviser pour régner » n'est pas suffisant pour conclure, mais l'utilisation de l'inégalité précédente pour n une puissance de 2 donne les inégalités successives

$$\begin{aligned}
C(0,n) &\le 2C(n/2,n) + M_{\mathbb{Z}}\Big(\log P(n/2,n)\Big)\\
&\le 2C(n/2,n) + M_{\mathbb{Z}}\Big(\frac{n}{2}\log n\Big)\\
&\le 4C(3n/4,n) + 2M_{\mathbb{Z}}\Big(\frac{n}{4}\log n\Big) + M_{\mathbb{Z}}\Big(\frac{n}{2}\log n\Big)\\
&\le 4C(3n/4,n) + 2M_{\mathbb{Z}}\Big(\frac{n}{2}\log n\Big)
\end{aligned}$$

$$\begin{aligned}&\leq \cdots\\ &\leq 2^k \mathrm{C}(n-2^{-k}n, n) + k\mathsf{M}_{\mathbb{Z}}\Big(\frac{n}{2}\log n\Big)\end{aligned}$$

pour tout entier positif k, où la quatrième ligne découle de la sous-additivité de la fonction $\mathsf{M}_{\mathbb{Z}}$. En choisissant finalement d'arrêter la récursion lorsque $n - 2^{-k}n$ et n diffèrent d'au plus un, ce qui impose k de l'ordre de $\log n$, on aboutit à la borne

$$\mathrm{C}(0, n) \leq \mathrm{O}(n) + \mathsf{M}_{\mathbb{Z}}\Big(\frac{n}{2}\log n\Big)\log n$$

donc à une complexité binaire pour le calcul de N! dans $\mathrm{O}\big(\mathsf{M}_{\mathbb{Z}}(\mathrm{N}\log \mathrm{N})\log \mathrm{N}\big)$.

Lorsque la multiplication entière utilisée repose sur la FFT, cette complexité devient $\mathrm{O}(\mathrm{N}\log^3 \mathrm{N}\log\log \mathrm{N})$; si la multiplication entière utilisée est d'exposant α strictement plus grand que 1, alors elle s'écrit $\mathrm{O}\big((\mathrm{N}\log \mathrm{N})^\alpha\big)$: au lieu d'utiliser simplement la sous-additivité de la multiplication, on majore la somme des $\mathsf{M}_{\mathbb{Z}}$ par une somme géométrique multipliée par le dernier sommant.

Récurrences d'ordre 1

La factorielle de la section précédente suit la récurrence $u_{n+1} = (n+1)u_n$. La méthode se généralise tout d'abord aux solutions de récurrences de la forme

$$u_{n+1} = p(n)u_n, \qquad p \in \mathbb{Z}[\mathrm{X}].$$

Si p a degré d, alors $\log p(\mathrm{N})$ est dans $\mathrm{O}(d\log \mathrm{N})$, si bien que la taille de u_{N} est au plus $\mathrm{O}(d\mathrm{N}\log \mathrm{N})$.

De même, pour toute récurrence de la forme

$$u_{n+1} = \frac{p(n)}{q(n)}u_n, \qquad p, q \in \mathbb{Z}[\mathrm{X}],$$

on peut pour calculer le terme u_{N} appliquer séparément la même technique de scindage binaire sur le numérateur et le dénominateur. Si d est le maximum des degrés de p et q, le calcul produit deux entiers de taille $\mathrm{O}(d\mathrm{N}\log \mathrm{N})$ en un nombre d'opérations binaires en $\mathrm{O}\big(\mathsf{M}_{\mathbb{Z}}(d\mathrm{N}\log \mathrm{N})\log \mathrm{N}\big)$.

À ce stade, on dispose d'un rationnel $u_{\mathrm{N}} = \mathrm{P/Q}$ dont le numérateur et le dénominateur sont des entiers potentiellement grands. Une évaluation numérique de ce rationnel à 2^{-e} près consiste à calculer $2^{-e}\lfloor 2^e \mathrm{P/Q}\rfloor$. La méthode de Newton (Théorème 4.2) permet de calculer l'inverse de Q à précision e en $\mathrm{O}(\mathsf{M}_{\mathbb{Z}}(e))$ opérations. Il est donc possible de terminer le calcul par une approximation de u_{N} à précision e jusqu'à $\mathrm{O}(d\mathrm{N}\log \mathrm{N})$ sans affecter la complexité. Cette observation est exploitée dans les sections suivantes. (Le raisonnement est le même dans une base différente de 2 ; le changement de base a un coût légèrement plus que linéaire.)

Calcul de $e = \exp(1)$

Le point de départ est la suite (e_n) donnée par $e_n = \displaystyle\sum_{k=0}^{n} \frac{1}{k!}$. Cette suite converge vers e. Plus précisément, il est classique que le terme de reste dans $e - e_n$ se majore par

une série géométrique de sorte que

$$0 \le e - e_n < \frac{1}{(n+1)!}\left(1 + \frac{1}{n+1} + \frac{1}{(n+1)^2} + \cdots\right) = \frac{1}{n\,n!}.$$

Pour calculer N décimales de e par cette série, il suffit de rendre $\frac{1}{n\,n!}$ inférieur à $10^{-\mathrm{N}}$. Il s'ensuit qu'il suffit de choisir N de sorte que $n\,n!$ et 10^{N} soient du même ordre, c'est-à-dire d'avoir N proportionnel à $n \log n$. Il suffit donc de $n = \mathrm{O}(\mathrm{N}/\log \mathrm{N})$ termes de la série.

La suite $(e_n)_{n\ge 0}$ vérifie $e_n - e_{n-1} = 1/n!$ et donc

$$n(e_n - e_{n-1}) = e_{n-1} - e_{n-2}.$$

Cette récurrence se récrit

$$\begin{pmatrix} e_n \\ e_{n-1} \end{pmatrix} = \frac{1}{n}\underbrace{\begin{pmatrix} n+1 & -1 \\ n & 0 \end{pmatrix}}_{\mathrm{A}(n)}\begin{pmatrix} e_{n-1} \\ e_{n-2} \end{pmatrix} = \frac{1}{n!}\mathrm{A}(n)\mathrm{A}(n-1)\cdots\mathrm{A}(1)\begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Le calcul du produit de matrices peut alors être effectué par scindage binaire. Le calcul de e_{N} par ce procédé demande donc $\mathrm{O}(\mathsf{M}_{\mathbb{Z}}(\mathrm{N}\log \mathrm{N})\log \mathrm{N})$ opérations binaires.

Pour calculer n décimales de e, la première étape ci-dessus construit deux entiers de taille $\mathrm{O}(\mathrm{N}\log \mathrm{N}) = \mathrm{O}(n)$ en $\mathrm{O}(\mathsf{M}_{\mathbb{Z}}(n)\log n)$ opérations binaires. La division finale ne demande que $\mathrm{O}(\mathsf{M}_{\mathbb{Z}}(n))$ opérations par l'algorithme de Newton. L'ensemble du calcul est donc quasi-optimal. (En outre, le $\log n$ n'est pas présent pour des multiplications comme celle de Karatsuba dont l'exposant de complexité est supérieur à 1.)

La généralisation de ces calculs à l'évaluation numérique de solutions d'équations différentielles linéaires est présentée au Chapitre 17.

Suites polynomialement récurrentes

L'Algorithme 15.2 traite le cas général d'une suite polynomialement récurrente vérifiant la récurrence (15.1) avec des coefficients polynomiaux p_i à coefficients entiers.

Théorème 15.3 Si N est un entier positif, d est une borne sur les degrés des polynômes $p_i \in \mathbb{Z}[n]$ coefficients de la récurrence (15.1) supposée régulière sur $\{0,\dots,\mathrm{N}\}$, et si ℓ est une borne sur la taille des coefficients des p_i telle la taille des numérateurs et dénominateurs des conditions initiales supposées rationnelles soient bornées par $\mathrm{O}(d\mathrm{N}\log \mathrm{N} + \ell \mathrm{N} + \mathrm{N}\log r)$, alors la complexité binaire de la méthode du scindage binaire pour calculer le N-ième terme d'une suite solution est

$$\mathrm{O}(r^{\theta}\mathsf{M}_{\mathbb{Z}}(d\mathrm{N}\log \mathrm{N} + \ell \mathrm{N} + \mathrm{N}\log r)\log \mathrm{N}),$$

où θ est un exposant réalisable pour la complexité du produit de matrices (voir Chapitre 8).

Entrée Une récurrence de la forme (15.1) avec $p_i \in \mathbb{Z}[X]$, $i = 0,\dots,r$, des conditions initiales rationnelles $u_0,\dots,u_{r-1}$, un entier N.
Sortie Les termes $u_{N-r+1},\dots,u_N$ de la solution.

1. Évaluer $A(N-r),\dots,A(1),A(0)$, où A est la matrice définie en page 271.
2. Calculer récursivement le produit $M_{N-r+1,0}$ de ces matrices par la formule

$$M_{i,j} = \begin{cases} A(j), & \text{si } i = j+1, \\ M_{i,k} \cdot M_{k,j}, & \text{avec } k = \lfloor (i+j)/2 \rfloor, \text{ sinon.} \end{cases}$$

3. Si $p_r(n) \neq 1$, calculer $w = p_r(N-r)\cdots p_r(0)$ par ce même algorithme ; sinon $w = 1$.
4. Renvoyer $\frac{1}{w} M \cdot {}^t(u_0,\dots,u_{r-1})$.

Algorithme 15.2 – Algorithme de scindage binaire.

Démonstration. La complexité du calcul de la matrice de passage est comme ci-dessus, puis le produit matrice vecteur avec les conditions initiales de taille $O(dN\log N + \ell N + N\log r)$ reste dans la complexité annoncée. ■

Comme précédemment, cette borne de complexité peut être améliorée à

$$O(r^\theta M_{\mathbb{Z}}(dN\log N + \ell N + N\log r))$$

si l'exposant α de la complexité $M_{\mathbb{Z}}(m) = O(m^\alpha)$ est supérieur à 1.

15.4 Récurrences singulières

Les résultats obtenus dans ce chapitre supposent que les récurrences sont régulières. L'extension de ces algorithmes aux récurrences singulières est possible, mais il faut d'abord préciser le problème en analysant les espaces de solutions qui apparaissent dans ce cas.

Espace des suites solutions

Lemme 15.4 L'ensemble des suites solutions de la récurrence

$$p_r(n)u_{n+r} + \cdots + p_0(n)u_n = 0 \tag{15.5}$$

pour $n \in \mathbb{N}$, forme un espace vectoriel sur $\mathbb{K}$.

Démonstration. La combinaison linéaire de solutions reste une solution, il s'agit donc bien d'un sous-espace vectoriel de $\mathbb{K}^{\mathbb{N}}$. ■

Dans le cas des récurrences à coefficients constants, la dimension de l'espace des solutions est exactement l'ordre de l'équation. Ici, la dimension de cet espace est plus délicate, comme le montre l'exemple suivant.

Exemple 15.1 Si la suite (u_n) obéit à la récurrence

$$n(n-1)u_{n+3} - (n-1)u_{n+2} + (n+1)u_{n+1} + 2nu_n = 0, \qquad n \in \mathbb{N},$$

alors les seules contraintes induites par la récurrence pour $n = 0$ et 1 sont

$$u_2 + u_1 = 0, \quad 2u_2 + 2u_1 = 0$$

alors qu'à partir de $n = 2$, la récurrence force la valeur de u_{n+3}. Autrement dit, les valeurs u_0, u_1, u_3, u_4 sont libres, et les autres s'en déduisent. La dimension de l'espace des solutions est donc 4 pour cette récurrence d'ordre 3.

Cette dimension dépend des racines entières du coefficient de tête p_r. Si

$$\mathrm{H} := \{h \in \mathbb{N} \mid p_r(h) = 0\} \quad \text{et} \quad \mathrm{J} := \{0, \dots, r-1\} \cup \{h + r \mid h \in \mathrm{H}\} \tag{15.6}$$

alors pour tout $n \in \mathbb{N} \setminus \mathrm{J}$, l'équation (15.5) détermine de manière unique u_n en fonction des valeurs précédentes. Les espaces vectoriels

$$\Lambda_{\mathrm{K}} := \left\{(u_0, \dots, u_n) \in \mathbb{K}^{n+1} \mid (15.5) \text{ est satisfaite pour } n \in \{0, \dots, \mathrm{K} - r\}\right\} \tag{15.7}$$

vérifient donc $\dim \Lambda_{\mathrm{K}} = \dim \Lambda_{\mathrm{K}-1}$ lorsque $\mathrm{K} \notin \mathrm{J}$.

Une façon simple de calculer la dimension de l'espace des solutions de la récurrence (15.5) ainsi qu'une base est donc de calculer la dimension et une base de Λ_{K} pour $\mathrm{K} = \max \mathrm{J}$. Les valeurs de u_{N} pour N plus grand peuvent ensuite être calculées efficacement pour des grandes valeurs de N par les Algorithmes 15.1 et 15.2 en observant que la récurrence obtenue en remplaçant n par $n - \max \mathrm{J}$ dans la récurrence (15.5) est régulière.

L'espace Λ_{K} est défini par $\mathrm{K} - r + 1$ équations linéaires (données par la récurrence) en $\mathrm{K} + 1$ inconnues (les valeurs de la suite), et les équations obtenues pour $n \notin \mathrm{J}$ sont linéairement indépendantes. Cette discussion permet de conclure :

Proposition 15.5 La dimension de l'espace des solutions de la récurrence (15.5) est comprise entre r et $r + \mathrm{card}(\mathrm{H})$, où $\mathrm{H} := \{h \in \mathbb{N} \mid p_r(h) = 0\}$.

Un calcul d'algèbre linéaire en dimension $\mathrm{K} := \max \mathrm{J}$ permet donc d'obtenir les K premières valeurs d'une base de l'espace des solutions. Dans de nombreux cas, la valeur de K n'est pas très grande, comme dans l'exemple ci-dessus, et cette approche est suffisante. Mais en général, le maximum de J a une valeur exponentielle en la taille binaire de la récurrence.

Exemple 15.2 Pour la récurrence

$$(n - 100)(n - 200)u_{n+1} - u_n = 0, \qquad n \in \mathbb{N},$$

la valeur de $\max \mathrm{J}$ est 201, alors que le plus grand des coefficients de la récurrence (20000) a une taille binaire de seulement 14 bits. En remplaçant 100 et 200 par N et 2N avec N grand, ces estimations deviennent $\sim 2\log \mathrm{N}$ pour la taille binaire de la récurrence, alors que $\max \mathrm{J}$ croît comme $2\mathrm{N}^2+1$. L'approche suggérée ci-dessus mène alors à la résolution d'un système linéaire de taille exponentielle par rapport à la taille binaire de la récurrence. Dans cet exemple, on peut procéder autrement : les espaces Λ_{N} définis en (15.7) sont de dimension 1 pour $\mathrm{N} \in \{0,\dots,100\}$, puis l'équation $-u_{100}=0$ obtenue pour $n=100$ force Λ_{101} a être à nouveau de dimension 1, avec un générateur tel que $u_0=\cdots=u_{100}=0$ et $u_{101}=1$. Les espaces suivants Λ_{N} pour $\mathrm{N} \in \{102,\dots,200\}$ sont encore de dimension 1, puis à nouveau l'équation $-u_{200}=0$ force l'espace Λ_{201} à être engendré par la suite $u_0=\cdots=u_{200}=0$, $u_{201}=1$, qui fournit la base de l'espace des suites solution de la récurrence. Pour la déterminer, il a donc suffi de résoudre deux systèmes linéaires de petite dimension en $n=100,200$. Cette idée est développée dans la section suivante pour faire chuter la complexité du calcul d'une telle base.

Conditions initiales généralisées

Définition 15.2 On appellera *conditions initiales généralisées* de la récurrence

$$p_r(n)u_{n+r}+\cdots+p_0(n)u_n=q(n) \tag{15.8}$$

pour le support $\{0,\dots,\mathrm{N}\}$ les valeurs u_i pour $i \in \mathrm{J} \cap \{0,\dots,\mathrm{N}\}$, où J est défini par (15.6). Lorsque le support est $\mathbb{N}$, il n'est pas mentionné.

Exemple 15.3 Pour une récurrence régulière, les conditions initiales généralisées sont les conditions initiales habituelles $u_0,\dots,u_{r-1}$.

Exemple 15.4 Pour la récurrence de l'Exemple 15.2, les conditions initiales généralisées sont u_0,u_{101},u_{201} pour les supports contenant $\{0,\dots,200\}$.

Exemple 15.5 Pour la récurrence de l'Exemple 15.1, $0,1,3,4$ sont les indices des conditions initiales généralisées pour les supports contenant $\{0,\dots,4\}$.

Ces conditions initiales généralisées fournissent un test effectif d'égalité de suites, d'après le Lemme 14.2. Un autre intérêt immédiat est qu'elles permettent un calcul des éléments de la suite par l'Algorithme 15.3. À titre de comparaison, le calcul des N premiers coefficients est donné par une variante de la Proposition 15.1, de même preuve.

Proposition 15.6 Étant donnée la récurrence (15.5) et des conditions initiales généralisées, l'Algorithme 15.3 calcule les N premiers éléments de la suite solution

Entrée Une récurrence de la forme (15.8), un entier $N \geq r-1$, des conditions initiales généralisées $u_0, \ldots, u_{r-1}, u_{j_1}, \ldots, u_{j_M}$ pour le support $\{0, \ldots, N\}$.
Sortie Les termes $u_0, \ldots, u_N$ de la solution.

1. Calculer les évaluations des coefficients $p_0(n), \ldots, p_r(n), q(n)$ en $n = 0, \ldots, N_r$.
2. Pour $n = 0, \ldots, N$ faire :
   a. Si n est l'indice d'une condition initiale généralisée alors u_n prend la valeur prescrite par cette condition initiale.
   b. Sinon, u_n est donné par (15.8) calculé avec les valeurs de l'étape (1) et les valeurs précédentes des u_i.
3. Renvoyer $u_0, \ldots, u_N$.

Algorithme 15.3 – Calcul naïf pour les récurrences singulières.

en

$$O\left(r \max(N, d) \frac{M(d)}{d}\right)$$

opérations arithmétiques, où d est une borne sur le degré des polynômes coefficients.

Si les coefficients de ces polynômes sont des entiers de taille bornée par ℓ et que les conditions initiales généralisées sont des rationnels de numérateurs et dénominateurs bornés par $O(dN \log N + \ell N + N \log r)$, alors la complexité binaire de ce calcul est

$$O\left(rN^2 M_{\mathbb{Z}}(d \log N + \ell + \log r)\right),$$

la taille du résultat étant $O\left(N^2(d \log N + \ell + \log r)\right)$.

Dans le cas de récurrences homogènes, les algorithmes rapides par « pas de bébés, pas de géants » et par scindage binaire s'étendent aussi au cas singulier, avec presque les mêmes complexités.

Proposition 15.7 Étant données la récurrence linéaire (15.5) et des conditions initiales généralisées, l'Algorithme 15.4 calcule le N-ième terme d'une suite solution en

$$O\left(r^\theta M(\sqrt{d(M+1)N}) \log(dN)\right)$$

opérations arithmétiques, où d est une borne sur le degré des coefficients de la récurrence, $M \leq d$ est le nombre de racines entières positives de p_r et θ un exposant réalisable pour la complexité du produit de matrices.

Si les coefficients des polynômes p_i sont entiers de taille bornée par ℓ et les conditions initiales généralisées sont des entiers de taille au plus $O(dN \log N + \ell N +$

Entrée Une récurrence de la forme (15.5), un entier $N \geq r-1$, des conditions initiales généralisées $u_0,\dots,u_{r-1},u_{j_1},\dots,u_{j_M}$ pour le support $\{0,\dots,N\}$.
Sortie Les termes $u_{N-r+1},\dots,u_N$ de la solution.

1. Si $N = r-1$ renvoyer $u_0,\dots,u_{r-1}$.
2. Si $N = j_k$ pour un certain k, calculer récursivement $u_{j_k-r},\dots,u_{j_k-1}$ par ce même algorithme ; renvoyer $u_{j_k-r+1},\dots,u_{j_k}$.
3. Soit $m = \max J \cap \{0,\dots,N\}$.
   a. Calculer récursivement $u_{m-r+1},\dots,u_m$.
   b. Appliquer l'un des Algorithmes 15.1 ou 15.2 pour calculer $u_{N-r+1},\dots,u_N$ avec la récurrence obtenue en changeant n en $n-m+r-1$ dans (15.5), et les conditions initiales $u_{m-r+1},\dots,u_m$.
   c. Renvoyer le résultat.

Algorithme 15.4 – Calcul rapide pour les récurrences singulières homogènes.

$N \log r)$, alors ce calcul utilise

$$O\Big(M_{\mathbb{Z}}(dN\log N + \ell N + N\log r)\Big(r^\theta \log N + r(M+1)\Big)\Big)$$

opérations binaires.

Démonstration. D'abord l'algorithme est correct par récurrence sur N : pour $N = r-1$, c'est évident ; pour N un élément de J, la valeur renvoyée est correcte si elle l'est pour $N-1$; et pour $N \notin J$, les valeurs $u_{m-r+1},\dots,u_m$ sont correctes par hypothèse de récurrence, et les conditions sont alors remplies pour invoquer les Algorithmes 15.1 ou 15.2 sur une récurrence régulière dans l'intervalle demandé.

Le résultat de complexité arithmétique est obtenu en invoquant l'Algorithme 15.1 pour chaque $j_k - 1$ avec $j_k < m$ et pour m, c'est-à-dire au plus $1+M$ fois, où $M = \operatorname{card} H$. À chaque appel, la récurrence obtenue en décalant les indices a des coefficients dont le degré d est inchangé. Le Théorème 15.2 donne une complexité arithmétique en $O(r^\theta M(\sqrt{dK})\log(dK))$ pour une étape calculant une différence d'indices K.

La somme de ces complexités est donc de la forme

$$O\Big(r^\theta\Big(M(\sqrt{dK_1})\log(dK_1) + \cdots + M(\sqrt{dK_m})\log(dK_m)\Big)\Big) =$$
$$O\Big(r^\theta \log(dN)\Big(M(\sqrt{dK_1}) + \cdots + M(\sqrt{dK_m})\Big)\Big) = O\Big(r^\theta \log(dN) M(\sqrt{dK_1} + \cdots + \sqrt{dK_m})\Big)$$

avec $dK_1 + \cdots + dK_m \leq dN$. La fonction racine carrée n'est pas convexe, mais l'inégalité

$$\sum_{i=1}^{n} \sqrt{x_i} \leq \sqrt{n}\sqrt{\sum_{i=1}^{n} x_i}$$

se prouve facilement en développant

$$\sum_{i=1}^{n}\left(\sqrt{x_i}-\frac{1}{n}\sum_{i=1}^{n}\sqrt{x_i}\right)^2\geq 0$$

et permet de borner la complexité par

$$\mathrm{O}\Big(r^{\theta}\mathsf{M}(\sqrt{md\mathrm{N}})\log(d\mathrm{N})\Big).$$

De la même manière, le résultat de complexité binaire est obtenu en invoquant l'Algorithme 15.2 pour chaque j_k-1 avec $j_k<m$ et pour m. À l'étape k, les polynômes coefficients de la récurrence décalée ont un degré inchangé, et des coefficients avec une taille majorée par $\ell+d\log j_{k-1}$ (avec $j_0=r-1$). La complexité binaire de l'ensemble des produits de matrices dans l'Algorithme 15.2 est alors donnée par le Théorème 15.3 comme

$$\mathrm{O}(r^{\theta}\mathsf{M}_{\mathbb{Z}}(d\mathrm{K}_k\log\mathrm{K}_k+(\ell+d\log j_{k-1})\mathrm{K}_k+\mathrm{K}_k\log r)\log\mathrm{K}_k).$$

En utilisant $j_{k-1}\leq\mathrm{N}$ et la sous-additivité de la fonction $\mathsf{M}_{\mathbb{Z}}$, ces complexités se somment en

$$\mathrm{O}(r^{\theta}\mathsf{M}_{\mathbb{Z}}(d\mathrm{N}\log\mathrm{N}+\ell\mathrm{N}+\mathrm{N}\log r)\log\mathrm{N}).$$

Il reste à considérer les produits matrice-vecteur de l'Algorithme 15.2, qui sont au plus au nombre de $\mathrm{M}+1$, et font intervenir des entiers de taille bornée par $\mathrm{O}(d\mathrm{N}\log\mathrm{N}+\ell\mathrm{N}+\mathrm{N}\log r)$, ce qui conclut la preuve. ■

Ces résultats, en $\tilde{\mathrm{O}}(\sqrt{\mathrm{N}})$ pour la complexité arithmétique, et en $\tilde{\mathrm{O}}(\mathrm{N})$ pour la complexité binaire, généralisent ceux du cas régulier, et sont à comparer à ceux de la Proposition 15.6 pour l'algorithme naïf.

Homogénéisation des récurrences inhomogènes

Lorsque la récurrence inhomogène (15.8) a un membre droit $q(n)$ lui-même polynomial, alors ses solutions sont aussi solutions de la récurrence homogène

$$q(n)\left(p_r(n+1)u_{n+1+r}+\cdots+p_0(n+1)u_{n+1}\right)-q(n+1)\left(p_r(n)u_{n+r}+\cdots+p_0(n)u_n\right)=0. \tag{15.9}$$

Cette récurrence a d'autres solutions que celles de la récurrence (15.8). Si J est l'ensemble des indices des conditions initiales généralisées pour l'équation inhomogène, alors celui de la récurrence homogénéisée est

$$\mathrm{J}':=\mathrm{J}\cup\{r\}\cup\{h+r+1\mid q(h)=0\}.$$

Les valeurs en $\mathrm{J}'\setminus\mathrm{J}$ permettant de retrouver exactement les mêmes solutions sont calculées facilement à partir de la récurrence inhomogène de départ.

Exemple 15.6 Voici une variante de l'Exemple 15.2 :

$$(n-100)(n-200)u_{n+1}-u_n=n-50,\qquad n\in\mathbb{N}.$$

Entrée Une récurrence de la forme (15.8), un entier $N \geq r-1$, des conditions initiales généralisées $u_0, \dots, u_{r-1}, u_{j_1}, \dots, u_{j_M}$ pour le support $\{0, \dots, N\}$.
Sortie Les termes $u_{N-r}, \dots, u_N$ de la solution.

1. Si $N = j_k$ pour un certain k, calculer récursivement $u_{j_k-r}, \dots, u_{j_k-1}$ par ce même algorithme ; renvoyer $u_{j_k-r+1}, \dots, u_{j_k}$.
2. Sinon, si $N \in J' := \{r\} \cup \{h \in \mathbb{N} \mid h > r \text{ and } q(h-r-1) = 0\}$ alors calculer récursivement $u_{N-r-1}, \dots, u_{j_k-1}$ par ce même algorithme (avec la convention $u_{-1} = 0$) ; calculer u_N par (15.8) ; renvoyer $u_{N-r}, \dots, u_N$.
3. Soit $m = \max(J \cup J') \cap \{0, \dots, N\}$.
   a. Calculer récursivement $u_{m-r}, \dots, u_m$.
   b. Appliquer l'un des Algorithmes 15.1 ou 15.2 pour calculer $u_{N-r+1}, \dots, u_N$ avec la récurrence obtenue en changeant n en $n-m+r$ dans la récurrence homogénéisée (15.9), et les conditions initiales $u_{m-r}, \dots, u_m$.
   c. Renvoyer le résultat.

Algorithme 15.5 – Calcul rapide pour les récurrences singulières inhomogènes.

L'équation homogénéisée correspondante est

$$(n-50)(n-99)(n-199)u_{n+2} - (n-50+(n-49)(n-100)(n-200))\,u_{n+1} + (n-49)u_n = 0.$$

Les indices des conditions initiales généralisées pour cette nouvelle récurrence sont $\{0, 1, 52, 101, 201\}$, c'est-à-dire ceux de la partie homogène, ainsi que $\{1, 52\}$. En ces indices, les valeurs fournies par la récurrence de départ sont :

$$u_1 = \frac{u_0 - 50}{20000}, \quad u_{52} = \frac{u_{51} + 1}{49 \times 149},$$

et la valeur de u_{51} peut quant à elle être obtenue par scindage binaire sur l'équation homogénéisée.

Procédant ainsi, l'Algorithme 15.5 permet de résoudre les équations inhomogènes dans la même complexité que précédemment.

Théorème 15.8 Étant données la récurrence linéaire inhomogène (15.8) et des conditions initiales généralisées, l'Algorithme 15.5 calcule le N-ième terme d'une suite solution en

$$O\left(r^\theta M(\sqrt{d(M+1)}N)\log(dN)\right)$$

opérations arithmétiques, où d est une borne sur le degré des coefficients de la récurrence, $M \leq 2d$ est le nombre de racines entières positives de q et p_r, et θ un

exposant réalisable pour la complexité du produit de matrices.

Si les coefficients des polynômes p_i et de q sont entiers de taille bornée par ℓ et les conditions initiales généralisées sont des entiers de taille au plus $O(d\mathrm{N}\log\mathrm{N} + \ell\mathrm{N} + \mathrm{N}\log r)$, alors ce calcul utilise

$$O\Big(\mathsf{M}_{\mathbb{Z}}(d\mathrm{N}\log\mathrm{N} + \ell\mathrm{N} + \mathrm{N}\log r)\Big(r^{\theta}\log\mathrm{N} + r(\mathrm{M}+1)\Big)\Big)$$

opérations binaires.

Démonstration. L'algorithme est correct par récurrence sur N, la seule différence avec la preuve de la Proposition 15.7 provenant des points de J′ qui sont soit des conditions initiales généralisées, soit des valeurs où la récurrence inhomogène permet le calcul.

L'analyse de complexité est exactement la même que pour le cas homogène. ■

Exercices

Exercice 15.2 — Calcul rapide de factorielle et de coefficients binomiaux centraux. Cet exercice montre comment calculer certaines suites récurrentes linéaires plus vite que par la méthode de scindage binaire. Soit $\mathrm{N} \in \mathbb{N}$ et soit $\mathrm{Q} = \sum_{i=0}^{2\mathrm{N}} q_i \mathrm{X}^i \in \mathbb{Z}[\mathrm{X}]$ le polynôme $\mathrm{Q}(\mathrm{X}) = (1+\mathrm{X})^{2\mathrm{N}}$.

1. Montrer que q_{N} peut être calculé en utilisant uniquement des additions d'entiers du triangle de Pascal, c'est-à-dire l'identité suivante sur les coefficients du binôme :
$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}, \qquad k \geq 1, n \geq 1.$$
Quelle est la complexité binaire de cet algorithme ?

On admet que le calcul de tous les nombres premiers inférieurs à N peut être effectué en $O(\mathrm{N}\log\mathrm{N}/\log\log\mathrm{N})$ opérations binaires, qu'il existe au plus $3\mathrm{N}/\log\mathrm{N}$ tels nombres premiers et que la multiplicité avec laquelle apparaît le nombre premier p dans la factorisation de N! vaut

$$\mathrm{ind}(p,\mathrm{N}) = \sum_{i=1}^{\infty} \left\lfloor \frac{\mathrm{N}}{p^i} \right\rfloor,$$

où la notation $\lfloor x \rfloor$ représente la partie entière de x.

2. Montrer que le calcul de $\mathrm{ind}(p,\mathrm{N})$ peut être effectué en
$$O\Big(\mathsf{M}_{\mathbb{Z}}(\log\mathrm{N})\log\mathrm{N}\Big)$$
opérations binaires.
3. Montrer que la décomposition en facteurs premiers de N! peut être effectuée en $O\Big(\mathsf{M}_{\mathbb{Z}}(\mathrm{N})\log\mathrm{N}\Big)$ opérations binaires, ainsi que celle de q_{N}.
4. Montrer que N! et q_{N} peuvent alors être reconstruits en un nombre d'opérations binaires bornés respectivement par $O\Big(\mathsf{M}_{\mathbb{Z}}(\mathrm{N}\log\mathrm{N})\log\log\mathrm{N}\Big)$ et $O\Big(\mathsf{M}_{\mathbb{Z}}(\mathrm{N})\log\mathrm{N}\Big)$.

5. Comparer avec la complexité du scindage binaire pour ces calculs.

■

Exercice 15.3 Soit $N \in \mathbb{N}$ et soit $P = \sum_{i=0}^{2N} p_i X^i \in \mathbb{Z}[X]$ le polynôme $P(X) = (1+X+X^2)^N$.

1. Montrer que $O(M(N))$ opérations binaires suffisent pour déterminer la parité de tous les coefficients de P.
   Indication : un entier n est pair si et seulement si $n = 0$ dans $\mathbb{K} = \mathbb{Z}/2\mathbb{Z}$.
2. Montrer que P vérifie une équation différentielle linéaire d'ordre 1 à coefficients polynomiaux. En déduire que les p_i suivent une récurrence d'ordre 2 que l'on précisera.
3. Donner un algorithme qui calcule p_N en $O(M(N \log N) \log N)$ opérations binaires.

■

Notes

L'algorithme par pas de bébés et pas de géants a été introduit par Strassen [Str77] et généralisé par les frères Chudnovsky [CC88] au problème du calcul d'un terme d'une suite récurrente linéaire à coefficients polynomiaux, dans un article où ils présentent également le scindage binaire et le doublement de précision.

L'algorithme de factorisation déterministe en Section 15.2 est basé également sur l'article de Strassen [Str77]. Cet algorithme peut à son tour être un peu amélioré [BGS07 ; CH14]. On ne connaît pas d'algorithme déterministe essentiellement meilleur ; l'existence d'un tel algorithme est un grand problème ouvert.

Une application cryptologique du calcul du N-ième terme d'une suite polynomialement récurrente au problème du comptage de points sur une courbe hyperelliptique sur un corps fini a été développée par Bostan, Gaudry et Schost [BGS07]. Le point de départ de cette application est l'exercice 15.1, inspiré de l'article de Flajolet et Salvy de 1997 [FS97, Pb. 4]. D'autres applications sont données au Chapitre 17.

La partie de l'Exercice 15.2 consacrée au calcul de N! est due à P. Borwein [Bor85].

Une conséquence des résultats de ce chapitre est que les N premiers coefficients d'une série différentiellement finie peuvent être calculés en complexité arithmétique $O(N)$ et en complexité binaire quasi-linéaire en N^2, en déroulant la récurrence linéaire à coefficients polynomiaux vérifiée par les coefficients de la série. Ces complexités sont quasi-optimales par rapport à la taille de la sortie.

En exploitant plus finement les mêmes récurrences sur les coefficients, on peut calculer le N-ième terme f_N d'une série différentiellement finie F plus vite que l'ensemble des $N+1$ premiers coefficients $f_0, \ldots, f_N$. Plus exactement, la technique des pas de bébés et pas de géants permet de calculer f_N en complexité arithmétique quasi-linéaire en $\sqrt{N}$ et la technique du scindage binaire de calculer f_N en complexité binaire quasi-linéaire en N. Cette dernière complexité est bien quasi-optimale en la taille de la sortie, mais celle en $\sqrt{N}$ ne l'est pas. Un grand problème ouvert est de savoir si l'on peut calculer f_N plus rapidement, ou si l'exposant 1/2 est une barrière intrinsèque. La réponse demeure à ce jour inconnue même dans des cas très particuliers, comme par exemple pour le calcul de $(\frac{p-1}{2})!$ mod p, pour un nombre premier p.

Un autre cas particulier important concerne les séries algébriques. Peut-on calculer le N-ième coefficient d'une série algébrique plus rapidement que celui d'une série

différentiellement finie quelconque ? La question vise les deux modèles de complexité, arithmétique et binaire. Par exemple, un tel coefficient a typiquement une taille binaire O(N), alors que le meilleur algorithme actuellement connu passe par le calcul intermédiaire d'une factorielle de matrices, de taille binaire $N \log N$. Dans le modèle arithmétique, aucun algorithme de complexité plus basse que $\sqrt{N}$ n'est actuellement connu, sauf dans le cas des corps finis, où une complexité en $\log N$ a été récemment obtenue [BCD16].

Bibliographie

BCD16 Bostan, Alin, Gilles Christol et Philippe Dumas (2016). « Fast Computation of the Nth Term of an Algebraic Series over a Finite Prime Field ». In : *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation*. ISSAC'16. ACM, p. 119–126.

BGS07 Bostan, Alin, Pierrick Gaudry et Éric Schost (2007). « Linear recurrences with polynomial coefficients and application to integer factorization and Cartier–Manin operator ». In : *SIAM Journal on Computing*, vol. 36, n°6, p. 1777–1806.

Bor85 Borwein, Peter B. (1985). « On the complexity of calculating factorials ». In : *Journal of Algorithms*, vol. 6, n°3, p. 376–380.

CC88 Chudnovsky, D. V. et G. V. Chudnovsky (1988). « Approximations and complex multiplication according to Ramanujan ». In : *Proceedings of the Centenary Conference, University of Illinois at Urbana-Champaign, June 1–5, 1987*. Éd. par G. E. Andrews, R. A. Askey, B. C. Berndt, K. G. Ramanathan et R. A. Rankin. Academic Press, p. 375–472.

CH14 Costa, Edgar et David Harvey (2014). « Faster deterministic integer factorization ». In : *Mathematics of Computation*, vol. 83, n°285, p. 339–345.

FS97 Flajolet, Philippe et Bruno Salvy (1997). « The SIGSAM challenges : symbolic asymptotics in practice ». In : *ACM SIGSAM Bulletin*, vol. 31, n°4, p. 36–47.

Str77 Strassen, V. (1976/77). « Einige Resultate über Berechnungskomplexität ». In : *Jahresbericht der Deutschen Mathematiker-Vereinigung*, vol. 78, n°1, p. 1–8.

16. Résolution de récurrences linéaires

Résumé

Les solutions polynomiales ou rationnelles de récurrences linéaires peuvent avoir une taille exponentiellement grande en la taille de l'équation. Néanmoins, elles s'obtiennent efficacement en utilisant la structure de leurs espaces de solutions, en évitant de développer les polynômes concernés, et en exploitant le calcul rapide de factorielles de matrices.

Les chapitres précédents montrent que de nombreuses informations concernant les solutions d'équations différentielles ou de récurrences linéaires peuvent être calculées directement à partir de l'équation sans passer par une « résolution » de celle-ci. Le cas particulier des solutions polynomiales et rationnelles mérite une attention spéciale : ces solutions, lorsqu'elles existent, permettent de nombreux calculs efficaces. D'autre part, décider de leur existence est une brique de base dans les algorithmes de sommation et d'intégration de la Partie VI.

Le cas le plus simple est celui d'équations *homogènes*, c'est-à-dire de la forme

$$p_r(n)u_{n+r} + \cdots + p_0(n)u_n = 0, \tag{16.1}$$

où les coefficients p_i sont des polynômes donnés à coefficients dans un corps $\mathbb{K}$ de caractéristique nulle. La question la plus générale que nous abordons est celle du calcul des solutions rationnelles, qui se ramène à celle des solutions polynomiales, elle-même se réduisant à la question des solutions à support fini (voir ci-dessous).

Un problème légèrement plus difficile, y compris du point de vue de la complexité, est celui des équations *inhomogènes*, de la forme

$$p_r(n)u_{n+r} + \cdots + p_0(n)u_n = q(n), \tag{16.2}$$

c'est-à-dire du même type, mais avec un membre droit qui n'est plus simplement 0, mais une suite $q(n)$ qui peut être à support fini, un polynôme ou une fraction rationnelle. Enfin, des algorithmes de sommation définie traités dans la Partie VI de cet ouvrage mènent à des équations *inhomogènes paramétrées*, où le membre droit est maintenant de la forme

$$q(n) = \lambda_1 q_1(n) + \cdots + \lambda_k q_k(n),$$

où les q_i sont des fractions rationnelles données, et les coefficients λ_i sont inconnus. Il s'agit alors de déterminer l'existence et la valeur des coefficients λ_i permettant l'existence d'une solution rationnelle de l'équation (16.2).

16.1 Suites solution

La forme d'une suite polynomialement récurrente

Dans le cas fréquent où les coefficients de la récurrence (16.1) sont des polynômes à coefficients entiers, les tailles des éléments de la suite croissent avec leur indice dans le cas général. Ces questions de taille ont été traitées au Chapitre 15. Il en ressort que la taille en bits du N-ième élément de la suite croît en $O(d\mathrm{N}\log \mathrm{N})$ où d est le degré des coefficients de la récurrence, et que la taille totale des N premiers éléments est bornée par $O(d\mathrm{N}^2 \log \mathrm{N})$.

Lorsque N est grand, ces tailles ont un impact sur les complexités, et il sera utile de concevoir des algorithmes qui opèrent sur ces suites sans nécessairement en représenter explicitement tous les éléments. En effet, les indices de suites pertinents dans les algorithmes de ce chapitre proviennent de récurrences rendues singulières par l'existence de solutions entières de polynômes. Ces solutions peuvent être exponentiellement grandes en la taille binaire de la récurrence (le polynôme $n - \mathrm{N}$ est un exemple frappant, voir aussi l'Exemple 15.2).

Calcul d'une base des solutions

Les Propositions 15.5, 15.7 et le Théorème 15.8 montrent qu'avec des conditions initiales généralisées en nombre au plus $r + \deg p_r (+ \deg q)$, il est possible de calculer, et ce efficacement, tout terme d'une suite solution d'une des récurrences (16.1) et (16.2). Un tel ensemble de conditions initiales généralisées fournit donc une structure de données pour représenter une solution. Tous les choix de conditions initiales généralisées ne correspondent pas nécessairement à une solution, mais il est possible de calculer une base des solutions dans cette représentation, tout en évitant des calculs dont la complexité arithmétique serait au moins linéaire en la plus grande racine entière du polynôme coefficient de tête p_r, ou du polynôme q dans le cas inhomogène.

Le principe du calcul est d'exploiter l'Algorithme 15.5 de calcule rapide par « pas de bébés, pas de géants » ou par scindage binaire avec des conditions initiales généralisées arbitraires pour en déduire les contraintes que l'évaluation de la récurrence aux zéros du polynôme p_r et à ceux du polynôme q dans le cas inhomogène fait porter sur ces conditions.

Entrée Une récurrence de la forme (15.5).
Sortie Les conditions initiales généralisées d'une base de ses solutions.

1. Calculer l'ensemble H des racines entières positives du coefficient de tête p_r.
2. En déduire $\mathrm{J} := \{0,\dots,r-1\} \cup \{h+r \mid h \in \mathrm{H}\}$.
3. Utiliser l'Algorithme 15.4 pour calculer des matrices $\mathrm{A}_1,\dots,\mathrm{A}_{\operatorname{card}\mathrm{H}}$ de dimensions $r \times \operatorname{card}\mathrm{J}$, où A_h avec $h \in \mathrm{H}$ contient pour colonnes les valeurs de $(u_h, u_{h+1},\dots,u_{h+r-1})$ pour les conditions initiales généralisées fournies par les colonnes successives de la matrice identité $\mathrm{I}_{\operatorname{card}\mathrm{J}}$.
4. Former une matrice M de dimensions $\operatorname{card}\mathrm{H} \times \operatorname{card}\mathrm{J}$ avec pour lignes
$$(p_0(h),\dots,p_{r-1}(h)) \cdot \mathrm{A}_h, \quad h \in \mathrm{H}.$$
5. Renvoyer une base du noyau de M et l'ensemble H.

Algorithme 16.1 – Résolution de récurrences homogènes singulières.

Exemple 16.1 La récurrence
$$(n-4)(n-8)u_{n+2} - u_{n+1} + u_n = 0, \qquad n \in \mathbb{N},$$
a $\{0,1,6,10\}$ comme ensemble d'indices pour ses conditions initiales généralisées. Les choix de valeurs (a,b,c,d) pour ces conditions initiales ne sont pas tous possibles, mais même avec des valeurs qui ne correspondent pas à des solutions, l'Algorithme 15.4 peut être exécuté, et permet de calculer
$$u_4 = \frac{5}{2016}a - \frac{13}{2016}b, \quad u_5 = \frac{1}{1260}a + \frac{1}{126}b,$$
$$u_8 = -\frac{1}{15120}a - \frac{1}{1512}b + \frac{1}{3}c, \quad u_9 = \frac{1}{9072}a + \frac{5}{4536}b - \frac{2}{9}c.$$
Les équations $u_4 - u_5 = 0$, $u_8 - u_9 = 0$ obtenues aux racines de $(n-4)(n-8)$ sont les seules contraintes qui pèsent sur les conditions initiales, qui sont donc fournies par le noyau de la matrice
$$\begin{pmatrix} \frac{17}{10080} & -\frac{29}{2016} & 0 & 0 \\ -\frac{1}{5670} & -\frac{1}{567} & 5/9 & 0 \end{pmatrix},$$
dont une base est donnée par $(1450,170,1,0),(0,0,0,1)$.

L'Algorithme 16.1 explicite cette idée dans le cas homogène. Lorsque la récurrence est régulière, l'étape (3) ne fait rien, la matrice de l'étape (4) est vide et la dernière étape renvoie la matrice identité I_r.

Récurrences inhomogènes

La récurrence linéaire inhomogène

$$p_r(n)u_{n+r} + \cdots + p_0(n)u_n = q(n), \quad n \in \mathbb{N} \tag{16.3}$$

a pour ensemble de solutions un espace affine : deux solutions diffèrent par un élément de l'espace vectoriel des suites solutions de son membre gauche. On peut voir ce problème comme un problème d'algèbre linéaire en dimension infinie : le vecteur (u_n) est multiplié par une matrice bande dont les coefficients de (n,n) à $(n,n+r)$ sont fournis par les évaluations des polynômes p_i en n ; ce produit doit être égal au vecteur $(q(n))$.

Le cas régulier où p_r n'a pas de racines entières signifie qu'il n'y a pas de 0 sur la diagonale de la matrice, et fournit donc une solution pour chaque condition initiale $(u_0,\dots,u_{r-1})$, que l'on obtient simplement en déroulant la récurrence.

Dans le cas général, il suffit de trouver des conditions initiales généralisées permettant à une solution d'exister jusqu'au plus grand zéro entier de p_r, puisqu'ensuite cette solution se prolonge de manière unique comme dans le cas régulier, et toutes les solutions sont obtenues en y ajoutant les solutions de l'équation homogène associée.

Exemple 16.2 La partie homogène de la récurrence

$$(n-4)(n-2)u_{n+1} - nu_n = 1$$

a un espace de solutions de dimension 2, engendré par les suites de conditions initiales généralisées $(u_0 = 1, u_2 = 0, u_4 = 0)$ et $(u_0 = 0, u_2 = 0, u_4 = 1)$. Par contre, la récurrence n'a pas de solution : en partant avec u_0 arbitraire, la récurrence produit

$$u_1 = \frac{1}{8}, \quad u_2 = \frac{3}{8}$$

puis la relation $-2u_2 - 1 = 0$, qui est incompatible avec ces équations.

Exemple 16.3 La récurrence

$$(n-4)(n-2)u_{n+2} + nu_{n+1} - u_n = 1$$

a une partie homogène admettant un espace de solutions de dimension 2. Les indices des conditions initiales généralisées sont $(0,1,4,6)$. Les premières évaluations de la récurrence fournissent les équations

$$8u_2 - u_0 = 1, \quad 3u_3 + u_2 - u_1 = 1, \quad 2u_3 - u_2 = 1, \quad -u_5 + 3u_4 - u_3 = 1, \quad 4u_5 - u_4 = 1,$$

puis des équations qui déterminent uniquement $u_7, u_8, \dots$ en fonction des précédents.

À ce stade, le problème est donc réduit à la résolution d'un système linéaire de dimension 6, et il reste à voir comment exploiter la structure supplémentaire induite par la récurrence. Pour cela on peut tout d'abord utiliser la partie homogène

Entrée Une récurrence de la forme (16.3).
Sortie Les conditions initiales généralisées d'une base de ses solutions.

1. Calculer la matrice M de l'Algorithme 16.1.
2. Utiliser l'Algorithme 15.5 avec des conditions initiales généralisées nulles pour calculer un vecteur B contenant les valeurs de la solution correspondante aux éléments de H.
3. Renvoyer une base des solutions de $MY + B = 0$ et l'ensemble H.

Algorithme 16.2 – Résolution de récurrences homogènes singulières.

de la récurrence avec des conditions initiales généralisées arbitraires en 0,1,4,6 ce qui donne pour $n = 1, 3$ les équations $(u_2, u_3) = (u_0/8, -u_0/24 + u_1/3)$ et $(u_4, u_5) = (u_4, u_0/24 - u_1/3 + 3u_4)$. Ensuite, on déroule la récurrence homogénéisée

$$(n-3)(n-1)u_{n+3} + (n+1-(n-4)(n-2))u_{n+2} - (n+1)u_{n+1} + u_n = 0$$

avec des conditions initiales généralisées nulles et la nouvelle condition en $n = 2$ fournie par l'équation inhomogène en $n = 0$, à savoir $u_2 = 1/8$. La récurrence homogène permet de calculer facilement (et efficacement même si ces indices étaient bien plus grands) les valeurs $(u_1, u_2, u_3) = (0, 1/8, 7/24)$, puis $(u_3, u_4, u_5) = (7/24, 0, -31/24)$. Par addition, le cas général vaut donc

$$(u_2, u_3) = (1/8 + u_0/8, 7/24 - u_0/24 + u_1/3), \quad (u_4, u_5) = (u_4, -31/24 + u_0/24 - u_1/3 + 3u_4).$$

Il ne reste plus qu'à introduire les équations pour $n = 2$ et 4 qui entraînent

$$-13/24 - 5u_0/24 + 2u_1/3 = -37/6 + u_0/6 - 4u_1/3 + 11u_4 = 0,$$

d'où finalement les conditions initiales

$$u_1 = 13/16 + 5u_0/16, \quad u_4 = 29/44 + u_0/44, \quad u_0 \text{ arbitraire.}$$

En résumé, la résolution d'un seul système linéaire de petite taille permet d'obtenir que la solution générale de l'équation a pour conditions initiales généralisées $(u_0, u_1, u_4, u_6) = (0, 13/16, 29/44, 0) + \lambda(1, 5/16, 1/44, 245/234344) + \mu(0, 0, 0, 1)$ avec λ, μ arbitraires.

L'Algorithme 16.2 précise cette méthode dans le cas général.

Racines entières

Sans information supplémentaire sur le corps $\mathbb{K}$ où se trouvent les coefficients des polynômes p_i de la récurrence, la recherche de racines entières peut poser des difficultés. Nous faisons donc l'hypothèse qu'un oracle les fournit sans compter ses opérations arithmétiques. Lorsque les coefficients sont entiers en revanche, les algorithmes de la Partie IV sur la factorisation peuvent être utilisés et donnent une complexité binaire

en $O(d^2\ell)$ pour trouver les racines entières d'un polynôme de degré d à coefficients de taille bornée par ℓ (grâce au Théorème 21.6 et aux Propositions 21.16 page 383 et 21.22 page 390).

Complexité

Théorème 16.1 L'ensemble H des racines entières positives du coefficient p_r (ainsi que celles de q dans le cas inhomogène) étant donné, l'Algorithme 16.1 (resp. 16.2) calcule les conditions initiales généralisées d'une base de solutions de la récurrence (16.1) (resp. (16.3)) en

$$O\Big(r^\theta\Big(M(\sqrt{d(M+1)N})\log(dN)+M(1+M/r)\Big)\Big)$$

opérations arithmétiques, où d est une borne sur le degré des coefficients de la récurrence, $M \le d$ (resp. $M \le 2d$) est le nombre de racines entières positives de p_r (resp. et de q) et N est une borne sur le plus grand élément de H.

Si de plus les coefficients de la récurrence sont dans $\mathbb{Z}[n]$ avec des coefficients de taille au plus ℓ, alors les conditions initiales généralisées peuvent être prises entières de taille $O(T)$ où $T = r(dN\log N + \ell N + N\log r)$, et ce calcul ainsi que celui de l'ensemble H (resp. et des racines de q) peut être effectué en

$$O\Big((r+M)^2 M_{\mathbb{Z}}(T)\log(T) + rMd^2 M_{\mathbb{Z}}(\ell+\log N)\Big)$$

opérations binaires.

Démonstration. D'après la Proposition 15.7, l'étape (3) de l'Algorithme 16.1 requiert $O\Big(r^\theta M(\sqrt{d(M+1)N})\log(dN)\Big)$ opérations arithmétiques pour calculer toutes les matrices de passage $r \times r$ nécessaires, qui sont indépendantes des conditions initiales. Il faut ensuite multiplier les M matrices ainsi construites par $\operatorname{card} J = r + M$ vecteurs de dimension r, que l'on peut grouper en $\operatorname{card} J/r$ matrices $r \times r$, ce qui amène le coût de l'ensemble de ces produits à $O(r^\theta M(1 + M/r))$ opérations arithmétiques supplémentaires.

L'étape suivante demande l'évaluation de chacun des polynômes coefficients en les éléments de H, ce qui, par évaluation multi-points, coûte $O(rMM(d)\log d/d)$ et est donc négligeable par rapport aux opérations précédentes. Dans le cas inhomogène, le Théorème 15.8 donne le vecteur B en la complexité annoncée. Enfin, la résolution du système linéaire coûte $O((M + r)r^{\theta-1})$ opérations arithmétiques d'après le Théorème 8.6, ce qui est du même ordre que les multiplications précédentes.

En ce qui concerne la complexité binaire, le calcul de l'étape (3) est donné par le Théorème 15.3 en $O\Big(r^\theta M_{\mathbb{Z}}(dN\log N + \ell N + N\log r)\log N)\Big)$ pour obtenir toutes les matrices de passage. Les vecteurs $(u_h, \dots, u_{h+r-1})$ ont une taille binaire en $O(dN\log N + \ell N + N\log r)$ ce qui fait que le calcul des matrices A_h est borné par celui des matrices de passage. Ces mêmes complexités apparaissent dans les étapes de l'Algorithme 15.5. Par les formules de Cramer et la borne d'Hadamard (Proposition 21.10), les vecteurs de la solution du système linéaire ont pour coordonnées des rationnels que l'on peut prendre de numérateur et dénominateur de taille $T = O(r(dN\log N + \ell N + N\log r))$.

Entrée Une récurrence linéaire régulière de la forme (16.1).
Sortie Une base des solutions à support fini, données par leurs conditions initiales.

1. Calculer $S := \{h \in \mathbb{N} \mid p_0(h) = 0\}$.
2. Si $S = \varnothing$, renvoyer $\varnothing$; sinon poser $N := \max S$.
3. Utiliser l'un des Algorithmes 15.1 ou 15.2 pour calculer la matrice de transition entre les indices $(0, \dots, r-1)$ et $(N+1, \dots, N+r)$.
4. Calculer une matrice C dont les colonnes forment une base du noyau de cette matrice.
5. Si le noyau est réduit à 0, renvoyer $\varnothing$; sinon renvoyer C.

Algorithme 16.3 – Calcul d'une base de solutions à support fini, cas régulier.

Leur calcul peut-être effectué en utilisant de l'algèbre linéaire modulo des nombres premiers en nombre de cet ordre en $O\left(r^\theta(1+M/r)T\right)$ opérations binaires, puis par reconstruction par restes chinois en $O((r+M)^2 M(T)\log T)$ opérations binaires, ce qui domine donc l'étape de construction des matrices de passage. ■

16.2 Solutions à support fini

Les algorithmes du reste de ce chapitre, pour calculer des solutions polynomiales ou rationnelles, se ramènent tous au calcul de solutions à support fini de récurrences linéaires.

Définition 16.1 On appelle *support* d'une suite (u_n), l'ensemble des $n \in \mathbb{N}$ tels que $u_n \neq 0$. Lorsque cet ensemble est fini, son plus grand élément s'appelle le *degré* de la suite, et on dit que le degré est infini sinon.

Les récurrences inhomogènes à coefficients polynomiaux comme (15.3) n'ont pas de solution à support fini : pour n plus grand que le degré, le membre gauche s'annule, et donc le membre droit, un polynôme, devrait avoir une infinité de racines, ce qui le force à être nul. Il suffit donc de considérer le cas homogène.

Proposition 16.2 Si (u_n) est une solution à support fini de la récurrence (16.1), alors son degré est racine du polynôme p_0. Réciproquement, si p_0 a K racines dans $\mathbb{N}$, alors la récurrence possède un espace de solutions à support fini de dimension comprise entre 1 et K.

Démonstration. Pour le sens direct, il suffit d'évaluer la récurrence en $n = N$ où N est le degré, ce qui donne $p_0(N)u_N = 0$. La réciproque est similaire à la Proposition 15.5 : l'évaluation de la récurrence en une racine entière h de p_0 ne contraint pas u_h, mais crée une contrainte linéaire sur les valeurs suivantes. Selon que cette contrainte est linéairement dépendante ou non des autres, elle permet ou non d'augmenter la dimension. ■

Entrée Une récurrence linéaire de la forme (16.1).
Sortie Une base des solutions à support fini, données par leurs conditions initiales généralisées.

1. Calculer $S := \{h \in \mathbb{N} \mid p_0(h) = 0\}$.
2. Si $S = \varnothing$, renvoyer $\varnothing$; sinon poser $N := \max S$.
3. Utiliser l'Algorithme 16.1 pour calculer une matrice B de dimension $\operatorname{card} J \times k$ dont les colonnes sont les conditions initiales généralisées d'une base de solutions.
4. Utiliser l'Algorithme 15.4 pour calculer une matrice de dimensions $r \times k$ dont les colonnes contiennent les valeurs aux indices $(N+1, \dots, N+r)$ de chacun des éléments de cette base.
5. Calculer une matrice C dont les colonnes forment une base du noyau de cette matrice.
6. Si le noyau est réduit à 0, renvoyer $\varnothing$; sinon renvoyer $B \cdot C$.

Algorithme 16.4 – Calcul d'une base de solutions à support fini, cas singulier.

Exemple 16.4 Voici un cas extrême où toutes les solutions sont à support fini :

$$u_{n+2} + (n-7)u_{n+1} + (n-10)(n-5)u_n = 0,$$

(exercice : le vérifier) mais changer le coefficient 7 en 6 par exemple fait chuter la dimension des solutions à support fini à 1, et le degré à 5.

Le principe de l'Algorithme 16.3 pour le cas de récurrences régulières est très simple : une fois trouvé N la plus grande racine entière positive du coefficient p_0, il s'agit de trouver les conditions initiales de la récurrence qui permettent d'obtenir $u_{N+1} = \cdots = u_{N+r} = 0$, ce qui donne l'ensemble des solutions à support fini dans ce cas. Dans le cas singulier, où le coefficient de tête de la récurrence a des zéros entiers qui peuvent à la fois augmenter la dimension de l'espace des solutions et ajouter des contraintes sur les conditions initiales, l'Algorithme 16.4, qui généralise le précédent, procède de façon similaire, à partir de l'Algorithme 16.1.

Théorème 16.3 Les racines entières positives de p_0 et p_r étant données, les Algorithmes 16.3 et 16.4 calculent une base de l'espace des solutions à support fini de la récurrence (16.1). Leur complexité arithmétique est

$$O\left(r^\theta M(\sqrt{dN}) \log(dN)\right)$$

où d est une borne sur le degré des coefficients et N la plus grande racine entière positive de $p_0 p_r$.

Lorsque les coefficients des polynômes p_i sont des entiers, la complexité binaire du calcul des racines entières positives de p_0 et p_r ainsi que du reste de l'algorithme

est
$$O\Big((r+M)^2 M_Z(T)\log(T) + rMd^2 M_{\mathbb{Z}}(\ell+\log N)\Big),$$
où $T = r(dN\log N + \ell N + N\log r)$ est alors une borne sur la taille des conditions initiales généralisées trouvées, ℓ étant une borne sur la taille des coefficients des polynômes p_i.

Démonstration. Les bornes de complexité des différentes étapes sont les mêmes que celles de l'Algorithme 16.1, la seule différence étant la borne N sur les indices. ■

R En pratique, même dans le cas où les coefficients sont entiers, si N est grand, il vaut donc mieux commencer par tester l'existence d'une racine modulo un nombre premier de taille modérée en un nombre d'opérations quasi-linéaire en $\sqrt{N}$, puis, s'il en existe, les calculer en un nombre quasi-linéaire d'opérations en N. Si l'on souhaite calculer tous les coefficients, alors avec ces conditions initiales généralisées, l'Algorithme 15.5 les fournit en complexité quasi-linéaire en N^2, ce qui est quasi-optimal. Pour bien des opérations cependant, ce calcul n'est pas nécessaire et la récurrence ainsi que ses conditions initiales généralisées fournissent une bonne structure de données grace aux algorithmes de cette partie.

16.3 Solutions polynomiales

Le principe est simple : on commence par trouver une borne sur le degré des solutions, puis la solution se ramène par coefficients indéterminés à un système linéaire. À nouveau, ce système a une structure induite par la récurrence linéaire, qu'il s'agit d'exploiter.

Borne sur le degré

Exemple 16.5 Si la récurrence
$$(n-3)u_{n+2} - (2n-3)u_{n+1} + nu_n = 0$$
a une solution qui se comporte pour n grand comme $u_n \sim n^k$, alors le membre gauche est un polynôme de degré $k+1$. L'extraction des coefficients de degré $k+1, k, k-1, \ldots$ fournit des équations qui doivent être satisfaites par une solution polynomiale. Il s'agit de
$$\begin{aligned} \text{degré } k+1 : &\quad 1-2+1 = 0, \\ \text{degré } k : &\quad 2k-3-2k+3 = 0, \\ \text{degré } k-1 : &\quad 2k(k-1) - 6k - k(k-1) + 3k = k(k-4) = 0, \end{aligned}$$
donc une solution polynomiale ne peut avoir que degré 0 ou 4.

Un algorithme pour ce calcul s'obtient un peu plus facilement en récrivant les décalages $u(n+k)$ de la suite initiale en termes de différences finies. Ainsi, on note $u(n)$ pour

u_n, $(\Delta\cdot u)(n) = u(n+1)-u(n)$ et, par récurrence, $(\Delta^{k+1}\cdot u)(n) = (\Delta^k\cdot u)(n+1)-(\Delta^k\cdot u)(n)$. La récurrence à annuler prend la forme

$$\mathrm{L}\cdot u = \sum_{k=0}^{m} b_k(n)\Delta^k u = 0$$

pour de nouveaux polynômes b_k. L'opérateur Δ fait décroître de 1 exactement le degré des polynômes. Ainsi,

$$\deg(\Delta^k \mathrm{P}) \leq \max(\deg \mathrm{P} - k, 0),$$

avec égalité tant que $\Delta^k \mathrm{P}$ n'est pas nul, et donc $\deg(\mathrm{L}\cdot \mathrm{P}) \leq \deg \mathrm{P} + \max_k\{\deg(b_k)-k\}$. Soient alors les quantités

$$b := \max_k\{\deg(b_k)-k\}, \qquad \mathrm{E} := \{k \mid \deg(b_k)-k = b\},$$

qui vont servir à fournir une borne sur le degré des solutions.

Exemple 16.6 La récurrence de l'exemple précédent se récrit

$$(n-3)(u_{n+2}-2u_{n+1}+u_n)+(2(n-3)-(2n-3))u_{n+1}+(3-n-3+n)u_n = 0,$$

qui se simplifie en

$$\Big((n-3)\Delta^2-3\Delta\Big)(u_n) = 0$$

l'entier b vaut 0 et l'ensemble E est $\{1,2\}$.

Soit D le degré d'une solution. La discussion distingue deux cas :

— soit $\mathrm{D}+b<0$, et alors $-(b+1)$ est une borne sur D ;

— sinon le coefficient de degré $\mathrm{D}+b$ dans $\mathrm{L}(n^{\mathrm{D}}+\cdots)$ vaut

$$\sum_{k\in \mathrm{E}} \mathrm{ct}(b_k)\mathrm{D}(\mathrm{D}-1)\cdots(\mathrm{D}-k+1),$$

où ct désigne le coefficient de tête. Cette expression, vue comme un polynôme en D, s'appelle le *polynôme indiciel* de la récurrence, lequel est non nul.

Cette discussion mène au résultat suivant.

Proposition 16.4 Une borne sur le degré des solutions polynomiales de l'opérateur $\mathrm{L} = \sum b_k(n)\Delta^k$ est donnée par le maximum de $-(b+1)$ et de la plus grande racine entière positive du polynôme indiciel de L.

Exemple 16.7 Dans l'exemple précédent, le polynôme indiciel vaut

$$\mathrm{D}(\mathrm{D}-1)-3\mathrm{D} = \mathrm{D}(\mathrm{D}-4),$$

ce qui redonne bien le résultat espéré.

Algorithme naïf

Un algorithme simple consiste alors à calculer cette borne et à rechercher ensuite les solutions par un calcul d'algèbre linéaire.

Exercice 16.1 Trouver les solutions polynomiales de la récurrence

$$3u(n+2) - nu(n+1) + (n-1)u(n) = 0.$$

■

Base binomiale

La borne sur le degré peut être exponentiellement grande en la taille binaire de la récurrence.

Exemple 16.8 La récurrence

$$nu(n+1) - (n+100)u(n) = 0$$

a pour solution le polynôme $u(n) = n(n+1)\cdots(n+99)$ de degré 100.

Comme dans les sections précédentes, il est alors souhaitable de disposer d'algorithmes permettant de détecter l'existence de solutions polynomiales en complexité maîtrisée par rapport à N. De plus, lorsqu'existent des solutions polynomiales de degré N, les développer utilise O(N) coefficients, alors qu'elles peuvent être représentées de manière compacte par la récurrence et les conditions initiales généralisées correspondantes.

Une difficulté provient de ce que les solutions polynomiales de récurrences linéaires ont des coefficients dans la base $1, n, n^2, \ldots$ qui ne vérifient pas une récurrence linéaire d'ordre borné a priori. En revanche, cette propriété apparaît si l'on développe les polynômes dans la base binomiale.

Proposition 16.5 Si (u_n) est une solution polynomiale de la récurrence (16.1) avec des coefficients de degré au plus d, alors ses coefficients (c_k) dans la base binomiale $\{\binom{n}{k}, k \in \mathbb{N}\}$ sont une suite solution à support fini d'une récurrence linéaire

$$a_m(k)c_{k+m} + \cdots + a_0(k)c_k = 0 \tag{16.4}$$

qui ne présente pas de nouvelles singularités, dont l'ordre est au plus $d + r$, les a_i ayant degré au plus d. Si les coefficients des p_i sont entiers de taille bornée par ℓ, ceux des a_i sont du même ordre.

Démonstration. Le cœur de la preuve se résume au calcul suivant.

Lemme 16.6 Si $u_n = \sum_{k=0}^{d} c_k \binom{n}{k}$ alors

$$nu_n = \sum_{k=0}^{d+1} k(c_k + c_{k-1})\binom{n}{k}, \qquad u_{n+1} = \sum_{k=0}^{d} (c_k + c_{k+1})\binom{n}{k}.$$

Démonstration. La relation

$$\binom{n}{k+1} = \frac{n-k}{k+1}\binom{n}{k}$$

se récrit en

$$n\binom{n}{k} = (k+1)\binom{n}{k+1} + k\binom{n}{k}.$$

Le premier résultat s'obtient en injectant cette récriture dans la somme nu_n et en extrayant le coefficient de $\binom{n}{k}$.

La seconde propriété découle directement du triangle de Pascal, qui s'obtient lui-même en extrayant le coefficient de X^k dans l'identité $(1+X)^{n+1} = (1+X)^n(1+X)$, ce qui donne

$$\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}.$$

■

En utilisant répétitivement ces deux propriétés, tout terme $n^i u_{n+j}$ apparaissant dans une récurrence se traduit en une portion de récurrence pour les coefficients, les multiplications par n pouvant augmenter l'ordre de 1, mais conservant le degré en k, tandis que les décalages en n ne changent pas le degré, et mènent à des décalages en k, d'où le résultat de la proposition en ce qui concerne les degrés et ordre. Enfin, on voit que le coefficient de tête de la récurrence est conservé dans la récurrence transformée, qui a donc les mêmes singularités. ■

Exercice 16.2 Montrer que l'existence d'une telle récurrence persiste pour toute base binomiale $\{\binom{n+a}{k}, k \in \mathbb{N}\}$ avec $a \in \mathbb{K}$ quelconque. ■

Exemple 16.9 Si (c_k) est la suite de coefficients de $\binom{n}{k}$ dans une solution polynomiale de la récurrence de l'Exemple 16.8, alors la traduction de cette récurrence montre que cette suite vérifie

$$k(c_k + c_{k+1} + c_{k-1} + c_k) - k(c_k + c_{k-1}) - 100c_k = kc_{k+1} + (k-100)c_k = 0,$$

ce qui donne une récurrence d'ordre 1 pour ces coefficients, et le degré 100 de la solution.

Exemple 16.10 La récurrence de l'Exemple 16.5 se traduit en

$$k(c_k + c_{k+1} + c_{k+1} + c_{k+2} + c_{k-1} + c_k + c_k + c_{k+1}) - 3(c_k + c_{k+1} + c_{k+1} + c_{k+2})$$
$$- 2k(c_k + c_{k+1} + c_{k-1} + c_k) + 3(c_k + c_{k+1}) + k(c_k + c_{k-1}) = (k-3)c_{k+2} + (k-3)c_{k+1} = 0,$$

équivalent à c_0 libre et $(k-4)c_{k+1} + (k-4)c_k = 0$ pour $k \geq 1$, où il apparaît clairement que $k = 4$ mène à une solution à support fini, de degré 4.

Conditions initiales

Comme pour les récurrences satisfaites par les coefficients de séries solutions d'équations différentielles linéaires (§14.1 page 253), l'ordre de la récurrence obtenue n'est pas nécessairement le même que celui de la récurrence initiale, et la traduction fournit également des contraintes linéaires sur les conditions initiales, qui précisent le sous-espace des solutions de la nouvelle récurrence correspondant aux solutions de celle de départ.

Exemple 16.11 La récurrence

$$v_{n+2} - v_{n+1} - (2n+1)v_n = 0$$

se traduit en

$$(c_k + 2c_{k+1} + c_{k+2}) - (c_k + c_{k+1}) - 2k(c_k + c_{k-1}) - c_k =$$
$$c_{k+2} + c_{k+1} - (2k+1)c_k - 2kc_{k-1} = 0$$

pour $k \geq 1$, et une contrainte obtenue en faisant $k = 0$ et $c_{-1} = 0$ dans cette équation, ce qui donne $c_2 + c_1 - c_0 = 0$. Ainsi la récurrence d'ordre 2 mène à une récurrence d'ordre 3, plus une contrainte linéaire sur les conditions initiales.

Conditions initiales généralisées

Si m est indice d'une condition initiale généralisée de la récurrence dont la suite (u_n) est solution, alors le lien entre les deux suites (u_n) et (c_k) donne

$$u_m = \sum_{k=0}^{m} c_k \binom{m}{k}$$

puisque les binomiaux $\binom{m}{k}$ sont nuls pour $k > m$. La condition initiale généralisée est donc donnée par

$$c_m = u_m - \sum_{k=0}^{m-1} c_k \binom{m}{k}.$$

Pour calculer cette valeur efficacement lorsque m est grand, il suffit de calculer d'abord une récurrence pour la suite $(c_k\binom{m}{k})_k$, par clôture (Thm. 14.6), puis pour la somme $(\sum_{k=0}^{i} c_k\binom{m}{k})_i$ en augmentant l'ordre de 1. La somme est alors calculée par l'Algorithme 15.4. Ainsi les conditions initiales généralisées sont calculées efficacement de proche en proche.

Entrée Une récurrence linéaire de la forme $p_r(n)u_{n+r} + \cdots + p_0(n)u_n = q(n)$.
Sortie Une base des solutions polynomiales, sous la forme d'une récurrence sur les coefficients des solutions dans la base binomiale, et de conditions initiales généralisées.

1. Calculer les conditions initiales généralisées d'une base des solutions par l'Algorithme 16.2.
2. Si la récurrence est inhomogène ($q \neq 0$), l'homogénéiser (§15.4 page 283).
3. Calculer la récurrence
$$a_m(k)c_{k+m} + \cdots + a_0(k)c_k = 0$$
donnée par la Proposition 16.5, ainsi que les équations linéaires sur les conditions initiales généralisées correspondant aux suites solutions de l'équation de départ.
4. Utiliser l'Algorithme 16.4 sur cette récurrence, en y adjoignant ces équations linéaires avant la résolution.
5. Renvoyer la récurrence et les conditions initiales généralisées de la base trouvée.

Algorithme 16.5 – Solutions polynomiales dans la base binomiale.

Solutions polynomiales

Grâce à cette traduction, la recherche de solutions polynomiales se ramène donc à la recherche de solutions à support fini. L'ensemble du calcul est résumé dans l'Algorithme 16.5. Les complexités arithmétiques et binaires découlent des considérations précédentes et sont résumées dans le théorème suivant.

Théorème 16.7 Les racines entières positives des coefficients a_0 et a_m de la récurrence (16.4) étant données, si N est la plus grande racine entière positive de a_0 et $\mathrm{M} \leq d$ une borne sur le nombre de racines entières positives de a_m inférieures à N, alors on peut déterminer les solutions polynomiales de la récurrence (16.1) en

$$\mathrm{O}\Big((r+d)^{\theta}\Big(\mathrm{M}(\sqrt{d(\mathrm{M}+1)\mathrm{N}})\log(d\mathrm{N}) + \mathrm{M}(1+\mathrm{M}/m)\Big)\Big)$$

opérations arithmétiques, sous la forme des conditions initiales généralisées d'une base de solutions exprimées dans la base binomiale.

Si de plus les coefficients p_i sont des entiers de taille bornée par ℓ, la complexité binaire du calcul de ces racines entières et du reste de l'algorithme est

$$\mathrm{O}\Big((r+d+\mathrm{M})^2\mathrm{M}_{\mathbb{Z}}(\mathrm{T})\log\mathrm{T} + (r+d)\mathrm{M}d^2\mathrm{M}_{\mathbb{Z}}(\ell+\log\mathrm{N})\Big),$$

où $\mathrm{T} = (r+d)(d\mathrm{N}\log\mathrm{N} + \ell\mathrm{N} + \mathrm{N}\log(r+d)$ est une borne sur les conditions initiales généralisées de la solution.

16.4 Solutions rationnelles

La recherche de solutions rationnelles se déroule en deux étapes : trouver d'abord un multiple du dénominateur, puis effectuer un changement de suite inconnue et chercher une solution polynomiale à la nouvelle équation ainsi obtenue pour le numérateur.

L'algorithme d'Abramov

Pour déterminer un multiple du dénominateur, il faut d'abord repérer les positions des pôles éventuels d'une solution rationnelle. L'intuition de départ s'obtient en considérant le cas de fractions dans le plan complexe. Si $u(n) = \mathrm{P}(n)/\mathrm{Q}(n)$ est solution de la récurrence

$$p_r(n)\frac{\mathrm{P}(n+r)}{\mathrm{Q}(n+r)} + \cdots + p_0(n)\frac{\mathrm{P}(n)}{\mathrm{Q}(n)} = q(n) \tag{16.5}$$

alors chaque racine a de $\mathrm{Q}(n)$ donne lieu à une racine $a-1$ de $\mathrm{Q}(n+1)$, une racine $a-2$ de $\mathrm{Q}(n+2)$ et ainsi de suite jusqu'à une racine $a-r$ de $\mathrm{Q}(n+r)$. Pourtant, le membre gauche de la récurrence doit rester un polynôme. Cela signifie que ces racines doivent correspondre à des racines des coefficients correspondants. Si aucune paire de racines de Q ne différait d'un entier, alors le dénominateur serait nécessairement un multiple du pgcd des polynômes $p_r(n-r), \ldots, p_1(n-1), p_0(n)$. La situation est plus complexe si deux racines de Q diffèrent d'un entier. Cependant, si α et β sont deux racines de Q telles que $\alpha - \beta = k \in \mathbb{N}$ est maximal, alors nécessairement $p_0(\alpha) = 0$ et $p_r(\beta - r) = 0$. La plus grande différence entière H entre les racines de Q peut donc être bornée étant donnés p_0 et p_r.

Cette observation peut être raffinée et abstraite du contexte du plan complexe. Soit

$$\mathrm{M}(n) = \operatorname{ppcm}(\mathrm{Q}(n+1), \ldots, \mathrm{Q}(n+r)).$$

La récurrence multipliée par $\mathrm{M}(n)$ s'écrit

$$p_r(n)\mathrm{P}(n+r)\frac{\mathrm{M}(n)}{\mathrm{Q}(n+r)} + \cdots + p_0(n)\mathrm{P}(n)\frac{\mathrm{M}(n)}{\mathrm{Q}(n)} = 0.$$

Dans cette récurrence, tous les coefficients $\mathrm{M}(n)/\mathrm{Q}(n+k)$, $k = 1, \ldots, r$ sont des polynômes. Par conséquent Q divise $p_0\mathrm{M}$, ce qui se récrit

$$\mathrm{Q}(n) \mid p_0(n) \operatorname{ppcm}(\mathrm{Q}(n+1), \ldots, \mathrm{Q}(n+r)).$$

En décalant n et en reportant, il vient

$$\begin{aligned}\mathrm{Q}(n) &\mid p_0(n) \operatorname{ppcm}(p_0(n+1) \operatorname{ppcm}(\mathrm{Q}(n+2), \ldots, \mathrm{Q}(n+r+1)), \mathrm{Q}(n+2), \ldots, \mathrm{Q}(n+r)) \\ &\mid p_0(n)p_0(n+1) \operatorname{ppcm}(\mathrm{Q}(n+2), \ldots, \mathrm{Q}(n+r+1)) \\ &\mid p_0(n)\cdots p_0(n+j) \operatorname{ppcm}(\mathrm{Q}(n+j+1), \ldots, \mathrm{Q}(n+j+r)).\end{aligned}$$

Cependant, pour $j > \mathrm{H}$, $\operatorname{pgcd}(\mathrm{Q}(n), \mathrm{Q}(n+j)) = 1$ et donc nous avons obtenu

$$\mathrm{Q}(n) \mid p_0(n)\cdots p_0(n+\mathrm{H}).$$

Entrée Une récurrence linéaire de la forme
$p_r(n)u_{n+r} + \cdots + p_0(n)u_n = q(n)$.
Sortie Un multiple du dénominateur des solutions rationnelles.

1. Calculer le polynôme
$$\mathrm{R}(h) = \mathrm{Res}_n\Big(p_0(n+h), p_r(n-r)\Big).$$
2. Si R n'a pas de racines dans $\mathbb{N}$, alors renvoyer le polynôme 1 ; sinon, soit $h_1 > h_2 > \cdots > h_m \geq 0$ ses racines entières positives. Initialiser Q à 1, A à $p_0(n)$, B à $p_r(n-r)$.
3. Pour $i = 1, \ldots, m$ faire
$g(n) := \mathrm{pgcd}\Big(\mathrm{A}(n+h_i), \mathrm{B}(n)\Big)$;
$\mathrm{Q}(n) := g(n)g(n-1)\cdots g(n-h_i)\mathrm{Q}(n)$;
$\mathrm{A}(n) := \mathrm{A}(n)/g(n-h_i)$;
$\mathrm{B}(n) := \mathrm{B}(n)/g(n)$.
4. Renvoyer Q.

Algorithme 16.6 – Algorithme d'Abramov pour les multiples du dénominateur.

De la même manière, multiplier le polynôme $\mathrm{ppcm}(\mathrm{Q}(n), \ldots, \mathrm{Q}(n+r-1))$ par la récurrence permet de voir que

$$\mathrm{Q}(n) \mid p_r(n-r)\cdots p_r(n-r-\mathrm{H})$$

et finalement

$$\mathrm{Q}(n) \mid \mathrm{pgcd}(p_0(n)\cdots p_0(n+\mathrm{H}), p_r(n-r)\cdots p_r(n-r-\mathrm{H}))$$

fournit un multiple de Q. Nous avons ainsi prouvé la première partie du résultat suivant.

Proposition 16.8 L'algorithme 16.6 calcule un multiple du dénominateur des éventuelles solutions rationnelles de la récurrence (16.1). Si les racines entières positives du polynôme $\mathrm{R}(h)$ sont données, sa complexité arithmétique est bornée par $\mathrm{O}(\mathsf{M}(d^2) + d\mathsf{M}(d)\log d)$ où d est une borne sur le degré des coefficients de la récurrence, à condition de ne pas développer les produits intervenant dans l'expression du polynôme renvoyé.

Démonstration. Le polynôme dans l'étape (1) est un résultant particulier qui peut se calculer efficacement comme une somme composée. Ce calcul est présenté dans le Chapitre 3 comme application du calcul rapide de l'exponentielle des séries. Sa complexité arithmétique est en $\mathrm{O}(\mathsf{M}(d^2))$.

Si on conserve le polynôme Q sous la forme

$$\mathrm{Q}(n) = \prod_i \prod_{j=0}^{h_i} g_i(n-j),$$

où g_i désigne le pgcd à l'étape i, le coût de la boucle est dominé par celui du pgcd. Tout d'abord, il faut décaler le polynôme A dont le degré reste borné par d, ce qui se fait à nouveau par une somme composée en $\mathrm{O}(\mathsf{M}(d))$ opérations arithmétiques. Puis chaque pgcd a une complexité arithmétique en $\mathrm{O}(\mathsf{M}(d)\log d)$ et le degré de B décroissant à chaque étape, le nombre d'étapes est au plus d, d'où le résultat. ■

L'estimation de la complexité binaire n'est pas compliquée, mais plus technique. Il faut borner les tailles des coefficients des différents polynômes et recourir au théorème des restes chinois pour les reconstruire efficacement.

Exercice 16.3 Trouver les dénominateurs possibles des solutions rationnelles des récurrences suivantes :

$$\begin{aligned}(n+1)u_{n+1} - nu_n &= 0,\\ (n+1)(n+3)u_{n+2} - 2(n+2)nu_{n+1} + (n-1)(n+1) &= 0,\\ (n+3)(n+2)(n^2+6n+4)u(n+2) - (n+1)(3n^3+27n^2+64n+48)u(n+1)&\\ + 2n^2(n^2+8n+11)u(n) &= 0.\end{aligned}$$

■

Exemple 16.12 Pour trouver les dénominateurs possibles pour les solutions rationnelles de la récurrence

$$(2n-1)(4n+4\mathrm{K}+5)(4n+4\mathrm{K}+3)u_{n+1} + 2(4n+5)(4n+3)(n+1)u_n = q(n),$$

où K est un entier strictement positif et q un polynôme, on commence par chercher les différences entières positives entre les racines de $(4n+5)(4n+3)(n+1)$ et $(2n-3)(4n+4\mathrm{K}+1)(4n+4\mathrm{K}-1)$, et il est facile de voir qu'il n'y a que $\mathrm{K}-1$. Le calcul par résultant peut se faire aussi, il s'agit de considérer

$$\mathrm{Res}_n(2(4n+4h+5)(4n+4h+3)(n+h+1), (2n-3)(4n+4\mathrm{K}+1)(4n+4\mathrm{K}-1)),$$

pour lequel on peut exploiter la multiplicativité du résultant (Cor. 6.6), ramenant ce calcul à des résultants de polynômes unitaires de degré 1, exhibant directement les différences de racines.

Il n'y a ensuite qu'un calcul de pgcd, qui extrait les facteurs menant à la racine commune $\mathrm{K}-1$: $g(n) = (4n+4\mathrm{K}+1)(4n+4\mathrm{K}-1)$. Un multiple des dénominateurs possibles est donc

$$\prod_{j=0}^{\mathrm{K}-1}(4n+4\mathrm{K}-4j+1)(4n+4\mathrm{K}-4j-1).$$

Numérateur

Une fois calculé ce multiple $\mathrm{Q}(n)$ du dénominateur, le changement d'inconnue $u(n) = \mathrm{P}(n)/\mathrm{Q}(n)$ dans l'équation (16.5) peut être réalisé efficacement, même si les h_i

sont très grands. C'est clair dans le cas où l'équation est homogène ($q(n) = 0$) puisqu'alors il suffit de multiplier le membre gauche par $\mathrm{Q}(n)$ et d'observer que

$$\frac{\mathrm{Q}(n)}{\mathrm{Q}(n+k)} = \prod_i \prod_{j=1}^{h_i} \frac{g_i(n-j)}{g_i(n+k-j)} = \prod_i \frac{g_i(n+k-h_i-1)g_i(n+k-h_i-2)\cdots g_i(n-h_i)}{g_i(n+k-1)g_i(n+k-2)\cdots g_i(n)}$$

est une fraction dont le numérateur et le dénominateur ont degré borné par

$$k \sum \deg(g_i) \leq k \max(\deg p_0, \deg p_r),$$

c'est-à-dire qui ne souffre pas de la taille potentiellement grande des h_i.

Exercice 16.4 Finir de trouver les solutions rationnelles pour les récurrences de l'Exercice 16.3. ■

Le cas inhomogène

Dans le cas inhomogène, ce changement d'inconnue mène à un membre droit de grand degré dépendant des h_i, mais celui-ci peut-être réduit : si le changement d'inconnue donne

$$\mathrm{A}(n, \mathrm{P}(n)) = \mathrm{Q}(n) \sum p_i(n) \frac{\mathrm{P}(n+i)}{\mathrm{Q}(n+i)} = \mathrm{Q}(n)q(n),$$

l'homogénéisation vue en §15.4 fournit une équation homogène linéaire

$$q(n)\mathrm{A}(n+1, \mathrm{P}(n+1)) - q(n+1)\frac{\mathrm{Q}(n+1)}{\mathrm{Q}(n)}\mathrm{A}(n, \mathrm{P}(n)) = 0 \tag{16.6}$$

dont les degrés des coefficients restent modérés. L'homogénéisation fournit aussi des équations sur les conditions initiales généralisées supplémentaires. Le problème est ainsi ramené à la recherche d'une base de solutions polynomiales, par l'Algorithme 16.5.

Le cas inhomogène paramétré

La récurrence (16.6) obtenue par homogénéisation d'une récurrence dont la partie inhomogène est $\lambda_1 q_1(n) + \cdots + \lambda_k q_k(n)$ produit *a priori* des divisions par des polynômes en les λ_i puisque $q(n)$ divise son coefficient de tête, mais les solutions restent linéaires en les λ_i et les conditions initiales en prenant en compte les contraintes linéaires sur les conditions initiales obtenues lors de l'homogénéisation.

Exemple 16.13 Pour trouver les valeurs de λ, μ permettant l'existence de solutions rationnelles à la récurrence

$$(2n-1)(2n-3)(4n+4\mathrm{K}+5)(4n+4\mathrm{K}+3)u_{n+1} + 2(4n+5)(4n+3)(2n-3)(n+1)u_n = 2\lambda(4n+5)(4n+3)(2n-3)(n+1) + 6\mu(4n+4\mathrm{K}+3)(4n+5)(n+1),$$

où K est un entier strictement positif, la première étape est de chercher un multiple des dénominateurs possibles, qui est fourni par la partie homogène. Une variante

simple de l'Exemple 16.12 donne le polynôme

$$\mathrm{Q}(n) := (2n-3)\prod_{j=0}^{\mathrm{K}-1}((4n+4\mathrm{K}-4j)^2-1).$$

Le facteur supplémentaire $(2n-3)$ peut soit être déduit du fait qu'il divise le membre gauche de la récurrence, soit obtenu automatiquement par l'Algorithme d'Abramov, qui fait apparaître 0 comme différence entière entre les coefficients.

Le changement d'inconnue $u_n = v_n/\mathrm{Q}(n)$ suivi par une multiplication par Q donne la récurrence

$$\begin{aligned}(2n-3)(4n+4\mathrm{K}+5)(4n+4\mathrm{K}+3)\frac{\prod_{j=0}^{\mathrm{K}-1}((4n+4\mathrm{K}-4j)^2-1)}{\prod_{j=0}^{\mathrm{K}-1}((4n+4\mathrm{K}+4-4j)^2-1)}v_{n+1}\\ +2(4n+5)(4n+3)(n+1)v_n =\\ (4n+5)(n+1)\,(2\lambda(4n+3)(2n-3)+6\mu(4n+4\mathrm{K}+3))\prod_{j=0}^{\mathrm{K}-1}((4n+4\mathrm{K}-4j)^2-1),\end{aligned}$$

où le facteur de v_{n+1} se simplifie en $(2n-3)(4n+3)(4n+5)$ et il ne reste donc des polynômes dont le degré dépend de K (donc potentiellement grands) que dans le membre droit. L'homogénéisation fournit alors une récurrence à coefficients de faible degré :

$$\begin{aligned}&(4n+5)(4n+7)(-1+2n)(n+1)\,((4n+3)(2n-3)\lambda+(12n+12\mathrm{K}+9)\mu)\,v_{n+2}\\ &-(n+2)\Big((4n+7)(2n-3)(16(2n-1)\mathrm{K}^2+32(2n^2+n-1)\mathrm{K}-48n^2-96n-45)\lambda\\ &\quad+3(4n+4\mathrm{K}+3)(16(2n-3)\mathrm{K}^2+8(2n^2-13)\mathrm{K}-80n^2-240n-175)\mu\Big)v_{n+1}\\ &-2(n+2)(n+1)(4n+4\mathrm{K}+5)(4n+4\mathrm{K}+3)\,((4n+7)(2n-1)\lambda+3(4n+4\mathrm{K}+7)\mu)\,v_n = 0,\end{aligned}$$

ainsi que la contrainte

$$-9v_1+6v_0 = (6\lambda-2(4\mathrm{K}+3)\mu)\,\mathrm{Q}(0)$$

sur les conditions initiales. Cette dernière est très importante, puisqu'elle permet de rester dans un sous-espace où les suites ont pour éléments des combinaisons linéaires de $\lambda, \mu, 1$ et non des fractions rationnelles en λ, μ. Par ailleurs, à ce stade, on observe que quitte à multiplier Q par un facteur constant, on peut prendre $\mathrm{Q}(0) = 1$.

Il faut ensuite chercher les solutions polynomiales de cette récurrence sur (v_n). La traduction en récurrence sur les coefficients dans la base binomiale mène à une récurrence d'ordre 7 un peu grosse pour être reproduite ici, ainsi que

des contraintes linéaires sur les conditions initiales, auxquelles il faut ajouter la contrainte

$$-9(c_0 + c_1) + 6c_0 = (6\lambda - 2(4\mathrm{K} + 3)\mu)\,\mathrm{Q}(0)$$

déduite de la précédente. Le coefficient de c_k dans la récurrence est le polynôme indiciel

$$512\lambda(k+5)(k+4)(k+3)(k+2)(k+1)(k-2\mathrm{K}-1)$$

dont la seule racine entière positive est $2\mathrm{K}+1$ qui est donc le seul degré possible pour une solution polynomiale.

La suite du calcul consiste donc à obtenir les équations sur c_0, λ, μ permettant l'existence d'une solution à support fini. Ces équations sont produites en utilisant la récurrence d'ordre 7 et selon le cas un calcul naïf ou de « pas de bébés-pas de géants » ou de scindage binaire pour obtenir (avec des choix $(c_0, \lambda, \mu) \in \{(0,1,0),(1,1,0),(0,0,1)\}$), les valeurs de $c_{2\mathrm{K}+2}, c_{2\mathrm{K}+3}, \ldots, c_{2\mathrm{K}+8}$ comme combinaisons linéaires de c_0, λ, μ. L'existence de solutions à ce système linéaire est équivalente à celle de solutions rationnelles à l'équation de départ.

Cette partie du calcul ne peut pas être effectuée avec K symbolique, mais par exemple avec K = 10, on obtient l'existence d'une solution avec

$$c_0 = 0, \quad \lambda = \frac{4790112341450}{13996552017}\mu.$$

À ce stade, tout dépend de ce que l'on souhaite calculer à propos de cette fraction rationnelle. Si l'objectif est d'obtenir les coefficients du numérateur, alors on peut utiliser la récurrence sur les (c_k) avec ces contraintes. On constate alors que le numérateur est un polynôme de degré 21, comme prédit par la borne, qui a un facteur de degré 1, le polynôme n, correspondant à $c_0 = 0$, et dont l'autre facteur est irréductible.

Exercice 16.5 Résoudre en (u, λ, μ) la récurrence

$$3u(n+2) - nu(n+1) + (n-1)u(n) = \lambda n^3 + \mu n^2.$$

■

Pour conclure, l'ensemble du calcul de solutions rationnelles, résumé par l'Algorithme 16.7, peut ainsi être réalisé efficacement, même si certains polynômes ont des degrés exponentiels en la taille de l'équation, à condition de ne pas développer ces polynômes, sous la forme d'un multiple du dénominateur exprimé comme un produit, et d'une récurrence avec ses conditions initiales généralisées pour une base des numérateurs exprimés dans la base binomiale.

L'ordre 1

Lorsque la récurrence est d'ordre 1, il est en outre possible de prédire un facteur du numérateur, ce qui gagne en efficacité pratique. En effet, dans la récurrence

$$a(n)u(n+1) + b(n)u(n) = c(n),$$

Entrée Une récurrence linéaire de la forme $p_r(n)u_{n+r} + \cdots + p_0(n)u_n = q(n)$, où $q(n)$ peut être nul, ou un polynôme, ou une combinaison linéaire de polynômes à coefficients λ_i inconnus.
Sortie Une base des solutions rationnelles, et éventuellement les valeurs des λ_i en permettant l'existence.
1. Calculer un multiple Q du dénominateur par l'Algorithme 16.6.
2. Poser $u_n = v_n/\mathrm{Q}(n)$ et former la récurrence qui en découle sur v_n ; éventuellement l'homogénéiser.
3. Calculer les solutions polynomiales de cette récurrence en procédant comme dans l'Algorithme 16.5 à partir de sa troisième étape.
4. Renvoyer Q, ces solutions et la récurrence correspondante.

Algorithme 16.7 – Solutions rationnelles de récurrences.

si b et c ont une racine commune qui n'est ni un pôle de u ni une racine de a, alors celle-ci est nécessairement racine de $u(n+1)$. De même, si a et c ont une racine commune qui n'est ni un pôle de $u(n+1)$ ni une racine de b, alors elle est racine de u.

Ces calculs préalables permettent de réduire le degré des coefficients de l'équation dont on recherche ensuite les solutions polynomiales. Ils sont utilisés dans l'algorithme de Gosper présenté au Chapitre 29.

Exercices

Ce chapitre montre qu'il est possible de déterminer les solutions à support fini de récurrences du type (16.1), puis par extension les solutions polynomiales de récurrences sous une *forme compacte* donnée par : la récurrence elle même à coefficients dans $\mathbb{Z}[\mathrm{X}]$ de degré borné par d ; une borne N sur le degré ; des conditions initiales généralisées (voir page 280) rationnelles de taille $\mathrm{O}(d\mathrm{N}\log\mathrm{N})$. Ceci n'est utile que si cette forme compacte se prête bien au calcul. L'exercice suivant et l'Exercice 17.4 du chapitre suivant explorent cette question.

Exercice 16.6 (Forme compacte en base binomiale.)
1. Montrer que si (u_k) est solution d'une récurrence linéaire de la forme (16.1) et si α n'est pas un entier négatif, la suite $(\sum_{i=0}^{k} u_i \binom{\alpha}{i})$ est solution d'une récurrence linéaire à coefficients polynomiaux que l'on peut calculer efficacement, ainsi que les r premières conditions initiales, dont le terme de tête n'a pas de nouvelles racines entières positives.
2. Montrer que si P est donné sous forme compacte dans la base binomiale, on peut calculer $\mathrm{P}(a)$ efficacement.
3. Même question pour le polynôme $\Delta^\ell \mathrm{P}(a)$ (ici Δ est l'opérateur aux différences est le polynôme $\Delta : \mathrm{P}(\mathrm{X}) \mapsto \mathrm{P}(\mathrm{X}+1) - \mathrm{P}(\mathrm{X})$, et Δ^ℓ représente l'itérée ℓ-ième de cet opérateur).

■

Notes

L'existence d'une récurrence pour les coefficients dans la base binomiale pour les solutions de récurrences linéaires est connue au moins depuis le 19e siècle [Boo72, Chap. XIII, art. 5]. Elle a été utilisée pour donner un algorithme calculant les solutions polynomiales par Abramov, Bronstein, Petkovšek [ABP95a]. Ensuite, ces questions ont été reprises du point de vue de l'efficacité en tenant compte de la taille potentiellement exponentielle des degrés en jeu, par l'introduction des algorithmes rapides pour la factorielle de matrice dans ce contexte, ainsi que les adaptations aux cas inhomogène et inhomogène paramétré. Ces idées, ainsi que les Exercices 17.4 et 16.6 proviennent d'articles de Bostan, Chyzak, Cluzeau, et Salvy [BCS05 ; Bos+06a].

Notre présentation de la preuve de l'algorithme d'Abramov est due à Chen, Paule et Saad [CPS08]. Abramov a proposé plusieurs raffinements de son algorithme pour obtenir des multiples du dénominateur de degré moindre [Abr95] et aussi dans certains cas en tenant compte de tous les coefficients p_i [Abr89]. Une manière plus directe d'aboutir à ces raffinements à été décrite par van Hoeij [Hoe98].

Bibliographie

ABP95a Abramov, Sergei A., Manuel Bronstein et Marko Petkovšek (1995). « On Polynomial Solutions of Linear Operator Equations ». In : *Symbolic and Algebraic Computation*. Éd. par A. H. M. Levelt. Proceedings of ISSAC'95, July 1995, Montreal, Canada. New York : ACM Press, p. 290–296.

Abr89 Abramov, S. A. (1989). « Rational solutions of linear differential and difference equations with polynomial coefficients ». In : *USSR Computational Mathematics and Mathematical Physics*, vol. 29, n°11, p. 1611–1620.

Abr95 — (1995). « Rational solutions of linear difference and q-difference equations with polynomial coefficients ». In : *ISSAC'95 : International Symposium on Symbolic and Algebraic Computation*. Éd. par A. H. M. Levelt. New York : ACM Press, p. 285–289.

BCS05 Bostan, Alin, Thomas Cluzeau et Bruno Salvy (2005). « Fast algorithms for polynomial solutions of linear differential equations ». In : *ISSAC'05 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 45–52.

Boo72 Boole, George (1872). *A treatise on the calculus of finite differences*. 2nd. Macmillan.

Bos+06a Bostan, A., F. Chyzak, T. Cluzeau et B. Salvy (2006). « Low complexity algorithms for linear recurrences ». In : *ISSAC'06 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 31–38.

CPS08 Chen, William Y. C., Peter Paule et Husam L. Saad (2008). « Converging to Gosper's algorithm ». In : *Advances in Applied Mathematics*, vol. 41, n°3, p. 351–364.

Hoe98 Hoeij, Mark van (1998). « Rational solutions of linear difference equations ». In : *ISSAC'98 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 120–123.

17. Résolution d'équations différentielles linéaires

Résumé

Comme pour les récurrences, les solutions polynomiales ou rationnelles d'équations différentielles linéaires peuvent avoir une taille exponentiellement grande par rapport à la taille de l'équation. Néanmoins, elles s'obtiennent efficacement en exploitant les algorithmes rapides sur les solutions de récurrences linéaires. Par ailleurs, la méthode de scindage binaire permet l'évaluation numérique garantie et rapide de solutions d'équations différentielles linéaires en complexité quasi-optimale vis-à-vis de la précision.

Comme au Chapitre 16, trois types d'équations sont considérés ici :

— homogènes, c'est-à-dire de la forme

$$c_m(x)y^{(m)}(x) + \cdots + c_0(x)y(x) = 0, \tag{17.1}$$

où les coefficients c_i sont des polynômes donnés à coefficients dans un corps $\mathbb{K}$;

— inhomogènes, c'est-à-dire du même type, mais avec un membre droit qui n'est plus simplement 0, mais un polynôme $b(x)$;

— inhomogènes paramétrées, où le membre droit est maintenant de la forme

$$\lambda_1 b_1 + \cdots + \lambda_k b_k,$$

où les b_i sont des polynômes donnés, et les coefficients λ_i sont inconnus.

Les solutions étudiées sont polynomiales, rationnelles, ou en séries. Les questions à aborder concernant ces solutions portent sur leur existence, leur valeur (et la structure de données à utiliser pour représenter cette valeur), la complexité arithmétique et binaire des calculs correspondants. En outre, dans le cas inhomogène paramétré, il s'agira de déterminer l'existence et la valeur des coefficients λ_i permettant l'existence d'une solution rationnelle de l'équation, cette opération intervenant dans les algorithmes de la Partie VI sur la sommation et l'intégration.

17.1 Séries solution

Traduction en récurrence

Le Théorème 14.1 page 253 montre que la suite des coefficients d'une série formelle solution de l'équation différentielle linéaire homogène (17.1) vérifie une récurrence linéaire homogène de type (16.1). Si les coefficients de l'équation différentielle sont

$$c_i(x) = \sum_j c_{i,j} x^j,$$

le calcul explicite de la récurrence vérifiée par la suite (y_n) des coefficients de la série $y(x)$ donne

$$\sum_{i,j} c_{i,j}(n-j+1)\cdots(n-j+i)y_{n+i-j} = 0, \tag{17.2}$$

pour tout $n \in \mathbb{Z}$, avec $y_\ell = 0$ pour $\ell < 0$. Quitte à décaler les indices, cette récurrence se récrit sous la forme

$$p_r(n)y_{n+r} + \cdots + p_0(n)y_n = 0, \tag{17.3}$$

à laquelle il faut ajouter les équations

$$p_r(-1)y_{r-1} + \cdots + p_0(-1)y_0 = 0, \quad p_r(-2)y_{r-2} + \cdots = 0, \ldots, \quad p_r(-r)y_0 = 0, \tag{17.4}$$

qui tiennent compte des contraintes $y_\ell = 0$ pour $\ell < 0$. (Voir l'Exemple 14.4 page 254.)

Définition 17.1 Le polynôme $p_r(n-r)$ est appelé *polynôme indiciel en 0* de l'équation (17.1).

Exercice 17.1 Le polynôme $p_0(n)$ est appelé polynôme indiciel à l'infini. En changeant la variable x en $x+\alpha$ dans l'équation, le même calcul fournit deux polynômes. Le premier s'appelle polynôme indiciel en α. Montrer que le polynôme indiciel à l'infini est inchangé. ■

Séries solutions

Proposition 17.1 Si $F \in \mathbb{K}[[x]]$ est solution de l'équation différentielle (17.1) alors sa valuation $\mathrm{val}\,F$ annule le polynôme indiciel en 0 de (17.1).

Démonstration. Il suffit d'évaluer la récurrence (17.3) en $n = \mathrm{val}(F) - r$. ■

Corollaire 17.2 Si le coefficient de tête c_m de l'équation différentielle (17.1) est non nul en 0, alors cette équation admet une base de m séries solutions ayant pour valuations $\{0, 1, \ldots, m-1\}$.

Démonstration. L'équation (17.2) montre que lorsque $c_{m,0} \neq 0$, le polynôme indiciel en 0 est donné par

$$p_r(n-r) = c_{m,0}(n-m+1)\cdots(n),$$

et donc la récurrence est régulière au sens du Chapitre 15. ■

Pour l'application importante au calcul des N premiers termes d'une série solution, on obtient donc.

Corollaire 17.3 Si le coefficient de tête c_m de l'équation différentielle (17.1) est non nul en 0, alors étant données ses m conditions initiales, on peut calculer les N premiers coefficients du développement en série d'une solution en

$$\mathrm{O}\left((m+d)\left(\mathsf{M}(d)\log d + \frac{\mathsf{M}(m)}{m}\mathrm{N}\right)\right)$$

opérations arithmétiques.

Si de plus les coefficients c_i de l'équation différentielle sont dans $\mathbb{Z}[x]$ avec des coefficients de taille bornée par ℓ, et les conditions initiales sont des rationnels de numérateurs et dénominateurs bornés par $\tilde{\mathrm{O}}(\mathrm{N}(m\log \mathrm{N} + d + \ell))$, alors la taille totale de ces N premiers coefficients est bornée par $\tilde{\mathrm{O}}(\mathrm{N}^2(m\log \mathrm{N} + d + \ell))$ et leur calcul peut s'effectuer en

$$\tilde{\mathrm{O}}\left((m+d)\mathrm{N}^2\mathsf{M}_{\mathbb{Z}}(m\log \mathrm{N} + d + \ell)\right)$$

opérations binaires.

Démonstration. La traduction d'équation différentielle en récurrence coûte

$$\mathrm{O}((m+d)\mathsf{M}(d)\log d)$$

opérations arithmétiques d'après le Théorème 14.5. D'après l'équation (17.3), la récurrence obtenue a ordre au plus $d+m$, degré au plus m, et ses coefficients, des combinaisons linéaires des coefficients des c_i avec des facteurs bornés par $(d+m)!$, ont donc taille en $\tilde{\mathrm{O}}(d+m+\ell)$ lorsque les $c_i \in \mathbb{Z}[x]$ ont des coefficients de taille au plus ℓ. La conclusion s'obtient par application de la Proposition 15.1. ■

Séries généralisées solutions

Les calculs ci-dessus s'étendent facilement à une classe un peu plus grande de solutions.

Définition 17.2 On appelle *série généralisée* une série de la forme $x^\alpha \mathrm{F}$, où $\mathrm{F} \in \mathbb{K}[[x]]$ et $\alpha \in \overline{\mathbb{K}}$.

Proposition 17.4 Si l'équation différentielle (17.1) admet une solution série généralisée alors α est racine du polynôme indiciel en 0 de (17.1). Si de plus, $\alpha \notin \{0, 1, \ldots, m-1\}$, alors $c_m(0) = 0$.

Démonstration. L'équation

$$x^{-\alpha}\left(c_m(x)(x^\alpha \mathrm{F})^{(m)} + \cdots + c_0(x)x^\alpha \mathrm{F}\right) = 0$$

est de même type que (17.1). La récurrence (17.3) qui lui correspond s'obtient en remplaçant n par $n-\alpha$ dans (17.3). Le résultat se déduit alors des deux résultats précédents. ■

Translation

Les solutions séries généralisées de la forme

$$y(t) = (t-a)^\alpha \sum_{i=0}^{\infty} y_i (t-a)^i \qquad \text{avec } y_0 \neq 0$$

se ramènent au cas précédent en effectuant le changement d'inconnue $y(a+t) = \tilde{y}(t)$ qui fournit l'équation

$$c_m(a+t)\tilde{y}(t)^{(m)} + \cdots + c_0(a+t)\tilde{y}(t) = 0. \tag{17.5}$$

Les résultats obtenus jusqu'ici établissent donc la proposition suivante.

Proposition 17.5 Si l'équation différentielle (17.1) admet une solution de la forme $(t-a)^\alpha \sum_{i=0}^{\infty} y_i (t-a)^i$ avec $y_0 \neq 0$ et $\alpha \notin \{0, \ldots, m-1\}$ alors $c_m(a) = 0$ et α est racine du polynôme indiciel de (17.5).

Définition 17.3 On appelle polynôme indiciel de l'équation différentielle (17.1) en a celui de l'équation (17.5) en 0.

Définition 17.4 On dit que $\alpha \in \overline{\mathbb{K}}$ est un point *ordinaire* de l'équation (17.1) si $c_m(\alpha) \neq 0$. On dit qu'il est *singulier* dans le cas contraire. On dit que l'infini est un point singulier de l'équation si 0 est singulier pour l'équation satisfaite par $y(1/x)$.

Exemple 17.1 La fraction rationnelle $y = 1/(1-x)$ est solution de l'équation

$$(1-x)y'(x) - y(x) = 0.$$

Le nombre complexe 1 est singularité de la solution et donc nécessairement point singulier de l'équation, ce qui se traduit par l'annulation du coefficient de tête.

Exemple 17.2 Le polynôme x^{10} est solution de l'équation

$$10xy'(x) - y(x) = 0.$$

La solution n'a pas de point singulier à distance finie, mais le nombre complexe 0 est un point singulier de l'équation ; le théorème de Cauchy ne s'y applique pas.

Preuves de non-finitude différentielle

Une conséquence utile de la Proposition 17.5 est que les fonctions différentiellement finies ne peuvent avoir qu'un nombre fini de singularités (les racines du coefficient de tête). Il s'en déduit des résultats négatifs.

Exemple 17.3 Les fonctions $1/\sin x$ et $\sqrt{\sin x}$ ne satisfont pas d'équation différentielle linéaire à coefficients polynomiaux.

Exemple 17.4 La suite des nombres de Bernoulli, qui interviennent en particulier dans la formule d'Euler–Maclaurin, ne peut être solution d'une récurrence linéaire à coefficients polynomiaux, puisque la série génératrice exponentielle

$$\sum_{n=0}^{\infty} \mathrm{B}_n \frac{z^n}{n!} = \frac{z}{\exp(z)-1}$$

a une infinité de pôles (aux points $2ik\pi$, pour $k \in \mathbb{Z} \setminus \{0\}$).

D'autres types de preuves de non-finitude différentielle sont présentés page 264.

17.2 Solutions polynomiales

Les solutions polynomiales de l'équation différentielle linéaire (17.1) ont des suites de coefficients à support fini solutions de la récurrence linéaire (17.3) qui s'en déduit par traduction, avec les contraintes sur les conditions initiales (17.4). Elles peuvent donc être calculées efficacement par les techniques du chapitre précédent. Les cas inhomogènes et inhomogènes paramétrés avec des membres droits polynomiaux passent à leurs analogues sur les suites dans cette traduction.

17.3 Solutions rationnelles

En un pôle a d'une solution rationnelle (c'est-à-dire un zéro de son dénominateur), la solution admet un développement en série généralisée de la forme $(t-a)^\alpha \sum y_i (t-a)^i$ avec $\alpha \in \mathbb{Z} \setminus \mathbb{N}$. D'après la Proposition 17.5, ces pôles sont donc faciles à localiser : ce sont des racines du coefficient de tête de l'équation différentielle, et leur multiplicité est bornée par l'opposé de la plus petite racine entière négative du polynôme indiciel en ces points.

Ceci conduit à l'Algorithme 17.1, essentiellement dû à Liouville, pour calculer les solutions rationnelles de (17.1). La preuve de l'algorithme se réduit à observer que le polynôme P qu'il calcule est un multiple du dénominateur de toute solution rationnelle. Du point de vue de la complexité, il faut noter que bien que les exposants N_a puissent avoir une valeur exponentielle en la taille de l'équation différentielle, le polynôme P n'a pas besoin d'être développé, et l'équation obtenue par changement de fonction inconnue grossit indépendamment des N_a, puisque la dérivée logarithmique de P est une fraction rationnelle avec un numérateur et un dénominateur de degré au plus d.

Entrée Une équation différentielle linéaire de la forme $c_m(x)y^{(m)}(x) + \cdots + c_0(x)y(x) = 0$.
Sortie Une base de ses solutions rationnelles.

1. En toute racine a de c_m :
   - calculer le polynôme indiciel $p_a(n)$;
   - calculer la plus petite racine entière négative N_a de p_a, s'il n'en existe pas, faire $\mathrm{N}_a := 0$.
2. Former le polynôme $\mathrm{P} = \prod_{c_m(a)=0}(x-a)^{-\mathrm{N}_a}$.
3. Effectuer le changement de fonction inconnue $y = \mathrm{Y}/\mathrm{P}$ et réduire au même dénominateur.
4. Chercher une base B des solutions polynomiales de cette nouvelle équation. Renvoyer la base des solutions rationnelles formée des fractions $\{b/\mathrm{P}, b \in \mathrm{B}\}$.

Algorithme 17.1 – Algorithme de Liouville pour les solutions rationnelles des équations différentielles linéaires.

Exercice 17.2 — Système et équation. Ramener la résolution du système

$$\mathrm{Y}'(x) = \mathrm{A}(x)\mathrm{Y}(x), \tag{17.6}$$

où $\mathrm{A}(x)$ est une matrice de fractions rationnelles de $\mathbb{K}(x)$ et Y un vecteur, à la résolution d'équations par la méthode de Liouville. ∎

Exercice 17.3 Trouver une équation différentielle linéaire satisfaite par y_1 solution de $y_1' = xy_1 - y_2$, $y_2' = y_1 - xy_2$. ∎

Le cas inhomogène

Lorsque le membre droit est un polynôme, le polynôme P trouvé par l'Algorithme 17.1 reste un multiple du dénominateur des solutions rationnelles. En revanche, l'étape de réduction au même dénominateur s'écrit

$$\mathrm{A}(x, \mathrm{Y}(x)) = \mathrm{P}(x) \sum_i c_i \frac{d^i}{dx^i} \frac{\mathrm{Y}(x)}{\mathrm{P}(x)} = \mathrm{P}(x)b(x)$$

et risque de faire exploser le degré des coefficients. Pour éviter cette explosion, il suffit de dériver encore une fois l'équation, et d'effectuer une combinaison linéaire :

$$b(x)\frac{d}{dx}\mathrm{A}(x, \mathrm{Y}(x)) - \left(b(x)\frac{\mathrm{P}'(x)}{\mathrm{P}(x)} + b'(x)\right)\mathrm{A}(x, \mathrm{Y}(x))$$
$$= b(x)(\mathrm{P}(x)b(x))' - \left(b(x)\frac{\mathrm{P}'(x)}{\mathrm{P}(x)} + b'(x)\right)\mathrm{P}(x)b(x) = 0.$$

Cette méthode s'applique aussi au cas inhomogène paramétré, et l'équation demeure linéaire en les paramètres.

17.4 Séries différentiellement finies analytiques

Le calcul de exp(1) présenté au chapitre précédent (§15.3) se généralise, jusqu'à l'évaluation numérique en bonne complexité de toutes les séries différentiellement finies convergentes dans une partie de $\mathbb{C}$.

Évaluation dans le disque de convergence

Soit $y(z)$ une solution de

$$a_r(z)y^{(r)}(z) + \cdots + a_0(z)y(z) = 0. \tag{17.7}$$

L'objectif est de calculer numériquement y à partir de conditions initiales. Le point de départ est un résultat d'existence, avec une preuve effective utile pour la suite.

Proposition 17.6 — Cauchy. Si $z_0 \in \mathbb{C}$ est tel que a_0 ne s'annule pas dans le disque $\mathcal{D}(z_0, \mathrm{R})$ centré en z_0 et de rayon R, alors pour tout $(y_0, \ldots, y_{r-1}) \in \mathbb{C}^r$, il existe une solution de (17.7) telle que $y^{(i)}(z_0) = y_i$ pour $0 \le i < r$ et qui est analytique dans $\mathcal{D}(z_0, \mathrm{R})$.

Démonstration. La preuve utilise une méthode de séries majorantes. L'équation (17.7) se récrit matriciellement

$$\mathrm{Y}' = \mathrm{AY}, \quad \mathrm{Y} = \begin{pmatrix} y \\ y' \\ \vdots \\ y^{(r-1)} \end{pmatrix}.$$

Comme le coefficient a_r ne s'annule pas dans $\mathcal{D}(z_0, \mathrm{R})$, les coefficients de la matrice A ont un développement en série convergeant dans ce disque. De manière équivalente, la matrice admet un développement en série

$$\mathrm{A}(z) = \sum_{k \ge 0} \mathrm{A}_k (z - z_0)^k,$$

tel que pour $0 < \rho < r$, il existe $\alpha > 0$ vérifiant $\|\mathrm{A}_k\| \le \alpha/\rho^{k+1}$ pour tout $k \ge 0$.

On considère alors la série formelle définie par

$$\mathrm{Y}(0) = \begin{pmatrix} y_0 \\ \vdots \\ y_{r-1} \end{pmatrix} \quad \text{et} \quad \mathrm{Y}' = \mathrm{AY},$$

c'est-à-dire dont les coefficients sont des vecteurs Y_n qui vérifient

$$(n+1)\mathrm{Y}_{n+1} = \sum_{k=0}^{n} \mathrm{A}_k \mathrm{Y}_{n-k}.$$

Des majorations directes donnent

$$(n+1)\|\mathrm{Y}_{n+1}\| \le \sum_{k=0}^{n} \|\mathrm{A}_k\|\|\mathrm{Y}_{n-k}\| \le \sum_{k=0}^{n} \frac{\alpha}{\rho^{k+1}} \|\mathrm{Y}_{n-k}\|.$$

Par récurrence, la suite (u_n) définie par $u_0 = \|Y_0\|$ et

$$(n+1)u_{n+1} = \sum_{k=0}^{n} \frac{\alpha}{\rho^{k+1}} u_{n-k}$$

est telle que $\|Y_n\| \le u_n$ pour tout n. Mais la série génératrice $u(z) = \sum u_n z^n$ vérifie

$$u'(z) = \frac{\alpha}{\rho - z} u(z),$$

d'où

$$u(z) = u_0 \left(1 - \frac{z}{\rho}\right)^{-\alpha}.$$

Comme les coefficients de la série Y sont majorés par ceux de u, qui converge, il s'ensuit que la série Y converge pour $|z - z_0| < \rho$. Et comme cette propriété est vraie pour tout $\rho < R$, la fonction Y est bien analytique dans $\mathcal{D}(z_0, R)$. ■

Cette preuve fournit aussi des bornes sur les queues du développement en série de $Y(z)$:

$$\begin{aligned}
\left\| \sum_{n \ge N} Y_n (z - z_0)^n \right\| &\le \sum_{n \ge N} u_n |z - z_0|^n, \quad \text{où} \quad u_n = u_0 \rho^{-n} \binom{\alpha + n - 1}{n}, \\
&\le u_0 \rho^{-N} \binom{\alpha + N - 1}{N} |z - z_0|^N \left(1 + \frac{\alpha + N}{1 + N} \frac{|z - z_0|}{\rho} + \cdots\right), \\
&\le C N^{\alpha} \left(\frac{|z - z_0|}{\rho}\right)^N,
\end{aligned}$$

pour une certaine constante C. Ainsi, pour garantir

$$\left\| \sum_{n \ge N} Y_n (z - z_0)^n \right\| \le 10^{-K},$$

il suffit de prendre

$$N \log_{10} \left(\frac{|z - z_0|}{\rho}\right) + \alpha \log_{10} N + \log_{10} C \le -K \tag{17.8}$$

ou encore

$$N \ge \frac{K}{\log_{10} \frac{\rho}{|z - z_0|}} + C' \log K \tag{17.9}$$

pour une certaine constante C'.

La combinaison de cette borne avec la méthode de scindage binaire du Chapitre 15 fournit donc une évaluation efficace de toutes les séries différentiellement finies à l'intérieur de leur disque de convergence.

Théorème 17.7 Si $y_i \in \mathbb{Q}$, $a_i \in \mathbb{Q}[z]$, $\zeta \in \mathbb{Q} \cap \mathcal{D}(z_0, \mathrm{R})$ ont tous des numérateurs et dénominateurs bornés par 2^ℓ, alors $y(\zeta)$ peut être calculé à précision $10^{-\mathrm{K}}$ en

$$\mathrm{O}\left(\mathsf{M}_{\mathbb{Z}}\left(\frac{\mathrm{K}(\ell+\log \mathrm{K})}{\log \frac{\rho}{|z-z_0|}}\right)\log \mathrm{K}\right)$$

opérations binaires, ainsi que $y'(\zeta), \ldots, y^{(r-1)}(\zeta)$.

Des informations plus précises sur la constante dans le O() et sa dépendance en le degré des coefficients et l'ordre de l'équation différentielle (17.7) s'obtiennent de la même manière sans difficulté autre que la taille des expressions à manipuler.

Démonstration. L'algorithme consiste dans un premier temps à obtenir une borne C comme dans le développement ci-dessus, puis à l'utiliser pour borner N grâce à l'équation (17.8). Cette borne croît au plus en $\mathrm{O}(\mathrm{K}/\log(\rho/|z-z_0|))$ d'après (17.9). Ensuite, il s'agit de calculer la somme du développement en série $y(z)$ au voisinage de z_0 jusqu'au N^{e} terme. Les sommants forment une suite polynomialement récurrente, donc la suite des sommes tronquées aussi. Comme z_0 est rationnel, les coefficients de la récurrence satisfaite par les coefficients sont des polynômes de $\mathbb{Q}[z]$. Cette récurrence est régulière par le Corollaire 17.2. La conclusion découle alors du Théorème 15.3. Le même raisonnement s'applique à une dérivée k^{e}, en remplaçant la borne u_n sur les coefficients par $n^k u_n$. À nouveau, l'inégalité (17.9) en découle, avec une constante C' éventuellement plus grande. ■

R L'énoncé ci-dessus est donné pour $\zeta \in \mathbb{Q}$ par simplicité, mais le même raisonnement s'applique à des points complexes du disque de convergence dont la partie réelle et la partie imaginaire sont rationnels. Plus généralement, il s'applique à des nombres algébriques dans le disque de convergence.

Prolongement analytique

L'équation différentielle (17.7) et ses conditions initiales étant données, il est possible d'étendre le raisonnement précédent pour calculer des valeurs de la solution en dehors du disque de convergence de son développement en série. Si les conditions initiales sont données en z_0, le principe est d'utiliser un point intermédiaire z_1 à l'intérieur du disque de convergence (borné par la racine la plus proche du coefficient de tête de l'équation). La méthode précédente permet de calculer de nouvelles conditions initiales en z_1. On calcule donc ainsi une *matrice de transition* permettant de passer de conditions initiales en z_0 à des conditions initiales en z_1. L'intérêt de ce calcul est que le disque centré en z_1 et borné par la racine la plus proche du coefficient a_r n'est pas nécessairement inclus dans le disque de départ, et permet donc d'atteindre de nouveaux points, où recommencer le calcul. Ensuite, en bornant les normes des matrices de transition, on peut borner les propagations d'erreurs et ainsi estimer les nombres de termes à prendre à chaque étape.

Doublement de précision

Le Théorème 17.7, combiné au prolongement analytique, montre comment calculer la valeur d'une fonction différentiellement finie à un point arbitraire de coordonnées rationnelles de taille modérée. Si le point ζ lui-même est donné à précision 10^{-K}, alors l'estimation obtenue par ce théorème avec $\ell = K \log_{10} 2$ donne une complexité quadratique. Il est heureusement possible d'obtenir une meilleure complexité par une technique de *doublement de précision*.

Théorème 17.8 Pour tout point $\zeta \in \mathcal{D}(z_0, R)$, la valeur $y(\zeta)$ peut être calculée à précision 10^{-K} en

$$O(K \log^3 K \log\log K)$$

opérations binaires étant donnés $O(K)$ bits de ζ.

Démonstration. Soient $z_0, z_1, \ldots, z_m = \zeta$ définis avec

$$z_i := \lfloor 2^{2^i} \zeta \rfloor 2^{-2^i},$$

c'est-à-dire que z_i est le nombre dyadique ayant les 2^i premiers bits de ζ. (On pourrait aussi raisonner en base 10, en considérant des nombres décimaux.) Le calcul de $y(z_{i+1})$ utilise donc des entiers plus gros que celui de $y(z_i)$, mais plus i est grand, plus $z_{i+1} - z_i$ est petit, et donc moins de termes de la série sont nécessaires. Ainsi, dans l'estimation de complexité du théorème, le terme

$$\frac{\ell + \log K}{\log \frac{\rho}{|z - z_0|}}$$

devient

$$\frac{2^i + \log K}{2^i + \log \rho},$$

et leur somme est

$$\sum_{i=1}^{\log K} \left(\frac{2^i}{2^i + \log \rho} + \frac{\log K}{2^i + \log \rho} \right) \leq \left(\sum_{i=1}^{\log K} 1 \right) + \sum_{i=1}^{\log K} \frac{\log K}{2^i} = O(\log K).$$

La formule finale est obtenue en utilisant pour simplifier la complexité de la multiplication d'entiers par FFT. ■

Exercice

Cet exercice est le pendant pour les équations différentielles de l'Exercice 16.6 du chapitre précédent sur l'utilisation de formes compactes pour le calcul efficace.

Exercice 17.4 (Forme compacte dans la base monomiale.)

1. Si le polynôme P est donné en forme compacte dans la base (X^k), c'est-à-dire au moyen d'une récurrence sur ses coefficients, et de conditions initiales généralisées, montrer comment obtenir efficacement son pgcd avec X^N.

2. Montrer que si (u_k) est solution d'une récurrence linéaire de la forme (16.1) et $\alpha \in \mathbb{Q} \setminus \{0\}$, alors la suite $(u_k \alpha^k)$ est solution d'une récurrence linéaire de la même forme, qui se calcule efficacement, dont le coefficient de tête a les mêmes racines entières positives que p_0, et dont les conditions initiales généralisées se calculent efficacement à partir de celles de (u_k).
3. Pour la suite $s_k := \sum_{i=0}^{k} u_i a^i$, montrer qu'on a encore facilement une récurrence linéaire, ainsi que les r premières conditions initiales, et que les racines du coefficient de tête sont encore inchangées.
4. Si (u_k) est donnée sous forme compacte, montrer comment obtenir les conditions initiales généralisées pour cette suite (s_k).
5. Montrer que si P est un polynôme donné sous forme compacte dans la base monomiale alors $P(\alpha)$ peut être calculé en un nombre d'opérations binaires quasi-linéaire en N, pour a comme ci-dessus.
6. Même question pour la dérivée $P^{(\ell)}(\alpha)$, pour $\ell = O(N)$.
7. Montrer comment obtenir les mêmes résultats pour $(\sum_{i=0}^{k} u_i \alpha^i)$ lorsque α est un nombre algébrique, donné par un polynôme $Q \in \mathbb{Q}[X]$ tel que $Q(\alpha) = 0$. (On admettra que les racines du terme de tête sont inchangées.)
8. En déduire que si P est donné en forme compacte dans la base (X^k), on peut en obtenir une forme compacte dans la base $((X - \alpha)^k)$ pour α donné par un polynôme $Q \in \mathbb{Q}[X]$ tel que $Q(\alpha) = 0$.
9. Décrire un algorithme permettant de diviser efficacement le numérateur obtenu sous forme compacte dans l'algorithme de Liouville par son pgcd avec le polynôme P multiple du dénominateur. Estimer sa complexité.

■

Notes

Les idées de base de l'algorithme de recherche de solutions rationnelles sont dues à Liouville [Lio33] qui donne également une méthode par coefficients indéterminés pour trouver les solutions polynomiales. La présentation qui utilise les récurrences donne un algorithme de même complexité mais fait mieux ressortir la structure du calcul [ABP95b]. Une bonne introduction aux équations différentielles dans le plan complexe est l'ouvrage classique de Ince [Inc56].

La recherche de solutions d'équations différentielles linéaires ne s'arrête pas aux solutions rationnelles. En utilisant la théorie de Galois différentielle, on peut concevoir une algorithmique sophistiquée de recherche de solutions *liouvilliennes* (c'est-à-dire formées par l'application répétée d'exponentielles, d'intégrales et de prise de racines de polynômes). Les calculs se ramènent à la recherche présentée ici de solutions rationnelles pour des équations (les puissances symétriques) formées à partir de l'équation de départ [Mar98 ; Sin81 ; SU97].

L'une des premières références mentionnant le scindage binaire [BGS72, §178] suggère déjà (sans preuve) qu'il permet l'évaluation numérique rapide des fonctions différentiellement finies. L'article de synthèse de Bernstein [Ber08, §12] est une bonne référence sur l'historique de la méthode.

La méthode de scindage binaire est un grand classique en calcul formel. Elle a été redécouverte à plusieurs reprises dans la littérature, dans différents contextes :

conversion d'une base à l'autre, calcul de décimales de constantes comme π ou e [Bre76], calcul avec des récurrences linéaires [KS73], [CC88, §6]. En particulier, c'est ainsi que les systèmes de calcul formel calculent rapidement π par une formule découverte en 1989 par les frères Chudnovsky :

$$\frac{1}{\pi} = \frac{12}{C^{3/2}} \sum_{n=0}^{\infty} \frac{(-1)^n (6n)!(A + nB)}{(3n)!n!^3 C^{3n}}$$

où A = 13591409, B = 545140134 et C = 640320. Cette série donne environ 14 décimales nouvelles par terme. Ceci n'est pas suffisant pour parvenir à une bonne efficacité, que l'on obtient en observant que les sommants vérifient une récurrence linéaire d'ordre 1 et en appliquant la méthode de scindage binaire.

Les algorithmes numériques présentés dans les dernières sections requièrent le calcul de bornes explicites pour le passage à l'implantation. C'est un travail un peu technique pour lequel nous renvoyons à la littérature [MS10].

Bibliographie

ABP95b Abramov, Sergei A., Manuel Bronstein et Marko Petkovšek (1995). « On polynomial solutions of linear operator equations ». In : *ISSAC'95 : International Symposium on Symbolic and Algebraic Computation*. Éd. par A. H. M. Levelt. New York : ACM Press, p. 290–296.

Ber08 Bernstein, Daniel J. (2008). « Fast multiplication and its applications ». In : *Algorithmic number theory : lattices, number fields, curves and cryptography*. Vol. 44. Publications of the Research Institute for Mathematical Sciences. Cambridge University Press, p. 325–384.

BGS72 Beeler, Michael, R. William Gosper et Richard Schroeppel (1972). *HAKMEM*. Artificial Intelligence Memo No. 239. Massachusetts Institute of Technology.

Bre76 Brent, Richard P. (1976). « Multiple-precision zero-finding methods and the complexity of elementary function evaluation ». In : *Analytic computational complexity*. Proceedings of a Symposium held at Carnegie-Mellon University, Pittsburgh, PA, 1975. Academic Press, p. 151–176.

CC88 Chudnovsky, D. V. et G. V. Chudnovsky (1988). « Approximations and complex multiplication according to Ramanujan ». In : *Proceedings of the Centenary Conference, University of Illinois at Urbana-Champaign, June 1–5, 1987*. Éd. par G. E. Andrews, R. A. Askey, B. C. Berndt, K. G. Ramanathan et R. A. Rankin. Academic Press, p. 375–472.

Inc56 Ince, E. L. (1956). *Ordinary differential equations*. Reprint of the 1926 edition. Dover Publications.

KS73 Kogge, P. et H. Stone (1973). « A parallel algorithm for the efficient solution of a general class of recurrence equations ». In : *IEEE Transactions on Computers*, vol. C-22, p. 786–793.

Lio33 Liouville, Joseph (1833). « Second mémoire sur la détermination des intégrales dont la valeur est algébrique ». In : *Journal de l'École polytechnique*, vol. 14, p. 149–193.

Mar98 Marotte, F. (1898). « Les équations différentielles linéaires et la théorie des groupes ». In : *Annales de la Faculté des Sciences de Toulouse*, vol. 12, n°4, p. 1–92.

MS10 Mezzarobba, Marc et Bruno Salvy (2010). « Effective Bounds for P-Recursive Sequences ». In : *Journal of Symbolic Computation*, vol. 45, n°10, p. 1075–1096.

Sin81 Singer, Michael F. (1981). « Liouvillian solutions of n-th order homogeneous linear differential equations ». In : *American Journal of Mathematics*, vol. 103, n°4, p. 661–682.

SU97 Singer, Michael F. et Felix Ulmer (1997). « Linear differential equations and products of linear forms ». In : *Journal of Pure and Applied Algebra*, vol. 117/118, p. 549–563.

IV Factorisation des polynômes

18. Factorisations sans carré et séparable

Résumé

Les algorithmes de factorisation des polynômes sont constitués de plusieurs étapes de natures bien différentes. Pour des polynômes à une variable, la première d'entre elles consiste à décomposer le polynôme en facteurs dont les racines sont de même multiplicité. On parle alors de factorisation séparable. Lorsque les coefficients appartiennent à un corps de caractéristique zéro ou plus grande que le degré, alors la factorisation séparable coïncide avec la factorisation sans carré. Le présent chapitre présente des algorithmes pour ces tâches.

18.1 Introduction

Soit $\mathbb{K}$ un corps, et soit F un polynôme unitaire de $\mathbb{K}[X]$. La *factorisation en irréductibles* de F, notée $\mathrm{firr}(F) = \{(F_1, m_1), \ldots, (F_r, m_r)\}$ avec les F_i irréductibles unitaires, correspond à l'identité

$$F = F_1^{m_1} \cdots F_r^{m_r}.$$

L'entier m_i est appelé la *multiplicité* du facteur F_i.

On dit que F est *sans carré* s'il n'existe pas de polynôme P de $\mathbb{K}[X] \setminus \mathbb{K}$ tel que P^2 divise F. Autrement dit, toutes les multiplicités des facteurs irréductibles de F sont égales à 1. Si l'on note $\{\mu_1, \ldots, \mu_s\}$ l'ensemble des multiplicités des facteurs irréductibles (avec $s \leq r$), et Q_i le produit des F_j de multiplicité $\mu_i = m_j$, alors la *factorisation sans carré* de F, notée $\mathrm{fsc}(F)$, est l'ensemble des paires (Q, m) où Q est le produit des facteurs irréductibles unitaires de F de multiplicité exactement $m \geq 1$. La *partie sans carré* de F est alors définie comme le produit $\prod_{(Q,m)\in \mathrm{fsc}(F)} Q$; il s'agit manifestement d'un polynôme sans carré.

Par exemple, si $\mathbb{K} = \mathbb{Q}$, et

$$F = (X+1)(X+2)(X+3)^2(X+4)^2(X+5)^3(X+6)^3,$$

sa partie sans carré est le produit de ses facteurs irréductibles, c'est-à-dire

$$(X+1)(X+2)(X+3)(X+4)(X+5)(X+6).$$

La factorisation sans carré de F est ici :

$$\mathrm{fsc}(F) = \{((X+1)(X+2),1),((X+3)(X+4),2),((X+5)(X+6),3)\}.$$

On voit bien que le calcul de factorisation sans carré est l'amorce de la factorisation irréductible ; en outre, c'est un calcul sensiblement plus facile. En effet, nous verrons plus loin un algorithme de calcul de la partie sans carré, de complexité quasi-linéaire en le degré, qui repose uniquement sur des calculs de pgcd bien choisis lorsque $\mathbb{K}$ est de caractéristique zéro ou suffisamment grande.

D'une façon générale, la factorisation irréductible d'un polynôme est l'une des premières questions difficiles du calcul formel. Nous présenterons ces difficultés et les solutions connues pour traiter les cas usuels, rencontrés en pratique, dans le Chapitre 22. Dans un premier temps nous exposerons les idées classiques en caractéristique zéro pour calculer la factorisation sans carré. En caractéristique non nulle la situation est nettement plus complexe et il existe des corps effectifs où l'extraction des racines p-ièmes n'est pas calculable. La factorisation sans carré n'est alors pas calculable en général et nous verrons alors l'intérêt d'introduire la factorisation séparable, qui nous sera utile dans les chapitres suivants.

Tout au long de ce chapitre, $\mathbb{K}$ représente un corps effectif, p sa caractéristique, et F un polynôme unitaire de $\mathbb{K}[X]$ de degré $n \geq 1$.

18.2 Factorisation sans carré

Dans cette section nous considérons le cas de la caractéristique zéro, et nous nous intéressons d'abord au calcul de la partie sans carré de F. Un algorithme pour la factorisation sans carré s'en déduit : on divise F par sa partie sans carré, on calcule la partie sans carré du résultat, etc. Ce faisant, on n'obtient pas une borne de complexité quasi-linéaire. L'algorithme rapide, dû à Yun, est présenté en fin de section.

Partie sans carré

Rappelons, qu'en caractéristique zéro un polynôme F de $\mathbb{K}[X]$ est sans carré si, et seulement si, $\mathrm{pgcd}(F, F') = 1$. Commençons par le lemme suivant, utilisé plusieurs fois dans la suite :

Lemme 18.1 Supposons $\mathbb{K}$ de caractéristique $p = 0$. Soient $G_1, \ldots, G_s$ des polynômes sans carré de $\mathbb{K}[X] \setminus \mathbb{K}$, premiers deux à deux, soit $G = G_1 \cdots G_s$, et soit

$(c_1,\dots,c_s)\in\mathbb{K}^s$. Alors nous avons l'égalité suivante :

$$\operatorname{pgcd}\left(\mathrm{G},\sum_{i=1}^{s}c_i\frac{\mathrm{G}}{\mathrm{G}_i}\mathrm{G}_i'\right)=\prod_{c_i=0}\mathrm{G}_i.$$

Démonstration. La preuve découle du calcul suivant :

$$\operatorname{pgcd}\left(\mathrm{G},\sum_{i=1}^{s}c_i\frac{\mathrm{G}}{\mathrm{G}_i}\mathrm{G}_i'\right)=\prod_{j=1}^{s}\operatorname{pgcd}\left(\mathrm{G}_j,\sum_{i=1}^{s}c_i\frac{\mathrm{G}}{\mathrm{G}_i}\mathrm{G}_i'\right)=\prod_{j=1}^{s}\operatorname{pgcd}\left(\mathrm{G}_j,c_j\frac{\mathrm{G}}{\mathrm{G}_j}\mathrm{G}_j'\right).$$

Ce dernier pgcd est égal à G_j si $c_j=0$ et 1 si $c_j\neq 0$ puisque G_j' est premier avec G_j. ■

Proposition 18.2 Si $\mathbb{K}$ est de caractéristique zéro, la partie sans carré de F est donnée par la formule $\mathrm{F}/\operatorname{pgcd}(\mathrm{F},\mathrm{F}')$ et peut être calculée au moyen de $\mathrm{O}(\mathsf{M}(n)\log n)$ opérations dans $\mathbb{K}$.

Démonstration. Écrivons la factorisation en irréductibles unitaires de F :

$$\mathrm{F}=\mathrm{F}_1^{m_1}\cdots\mathrm{F}_r^{m_r},$$

où tous les m_i sont des entiers ≥ 1. En dérivant, on obtient

$$\mathrm{F}'=\sum_{i=1}^{r}m_i\mathrm{F}_i'\frac{\mathrm{F}}{\mathrm{F}_i}.$$

En posant $\mathrm{E}=\prod_{i=1}^{r}\mathrm{F}_i$ la partie sans carré de F, on en déduit que

$$\operatorname{pgcd}(\mathrm{F},\mathrm{F}')=\operatorname{pgcd}\left(\prod_{i=1}^{r}\mathrm{F}_i^{m_i},\sum_{i=1}^{r}m_i\frac{\mathrm{F}}{\mathrm{F}_i}\mathrm{F}_i'\right)=\operatorname{pgcd}\left(\mathrm{E},\sum_{i=1}^{r}m\frac{\mathrm{E}}{\mathrm{F}_i}\mathrm{F}_i'\right)\prod_{i=1}^{r}\mathrm{F}_i^{m_i-1}=\prod_{i=1}^{r}\mathrm{F}_i^{m_i-1},$$

où la dernière égalité provient du Lemme 18.1. La partie sans carré E de F s'obtient alors comme $\mathrm{F}/\operatorname{pgcd}(\mathrm{F},\mathrm{F}')$. Le coût du calcul est essentiellement une conséquence de la Proposition 6.14 qui donne le coût du pgcd. ■

Algorithme de Musser

La première méthode de factorisation sans carré, que nous présentons dans l'Algorithme 18.1, commence par calculer la partie sans carré.

Proposition 18.3 L'Algorithme 18.1 de factorisation sans carré est correct et son coût arithmétique est en $\mathrm{O}(n\mathsf{M}(n)\log n)$.

Démonstration. Notons $(\mathrm{G}_1,m_1),\dots,(\mathrm{G}_s,m_s)$ la factorisation sans carré de F normalisée de sorte que

$$\mathrm{F}'=\sum_{i=1}^{s}m_i\mathrm{G}_i'\mathrm{G}_i^{m_i-1}\frac{\mathrm{F}}{\mathrm{G}_i^{m_i}}.$$

Entrée F dans $\mathbb{K}[X]$, avec $\mathbb{K}$ de caractéristique zéro.
Sortie fsc(F).
1. Initialiser L avec la liste vide et i à l'entier 1.
2. Calculer $C = \mathrm{pgcd}(F, F')$ et $G = F/C$.
3. Tant que $\deg(G) \geq 1$ faire :
   a. calculer $P = \mathrm{pgcd}(C, G)$ et $C = C/P$;
   b. si $\deg(G) > \deg(P)$ alors adjoindre $(G/P, i)$ à L ;
   c. $G = P$, et $i = i + 1$.
4. Renvoyer L.

Algorithme 18.1 – Algorithme de Musser en caractéristique zéro.

Entrée F dans $\mathbb{K}[X]$, avec $\mathbb{K}$ de caractéristique zéro.
Sortie fsc(F).
1. Poser $l = 1$ et initialiser L avec la liste vide.
2. Calculer $U = \mathrm{pgcd}(F, F')$, $V = F/U$, et $W = F'/U$.
3. Tant que $\deg(V) \geqslant 1$ faire :
   a. calculer $H = \mathrm{pgcd}(V, W - V')$, $W = (W - V')/H$, et $V = V/H$;
   b. si $\deg(H) \geqslant 1$ alors adjoindre (H, l) à L ;
   c. incrémenter l de 1.
4. Renvoyer L.

Algorithme 18.2 – Algorithme de Yun en caractéristique zéro.

Alors, à l'étape (2), nous avons $C = \prod_{i=1}^{s} G_i^{m_i - 1}$, et $G = \prod_{i=1}^{s} G_i$. Dans la première étape de la boucle nous obtenons $P = \prod_{i=1, m_i \geq 2}^{s} G_i$, $C = \prod_{i=1, m_i \geq 2}^{s} G_i^{m_i - 2}$, et G/P correspond aux G_i pour lesquels $m_i = 1$. À la deuxième étape de la boucle nous avons

$$P = \prod_{i=1, m_i \geq 3}^{s} G_i, \quad C = \prod_{i=1, m_i \geq 3}^{s} G_i^{m_i - 3},$$

et G/P correspond aux G_i pour lesquels $m_i = 2$. Par récurrence nous déduisons que l'algorithme est correct.

Le nombre d'étapes de l'algorithme est borné par n, le coût de chaque étape est dominé par celui du pgcd qui est, rappelons-le, en $O(M(n) \log n)$. ■

Algorithme de Yun

La méthode suivante de factorisation sans carré, présentée dans l'Algorithme 18.2, permet d'atteindre un coût arithmétique essentiellement linéaire.

Proposition 18.4 L'Algorithme 18.2 de factorisation sans carré est correct et son coût arithmétique est en $O(M(n) \log n)$ opérations dans $\mathbb{K}$.

Démonstration. Notons V_l et W_l les valeurs respectives de V et W au départ de l'étape l de la boucle « tant que », et notons H_l la valeur de H calculée durant l'étape l. Notons ℓ la valeur de l à la sortie de la boucle. Définissons aussi V_ℓ et W_ℓ comme les valeurs respectives de V et W à la sortie de la boucle. Nous allons prouver la propriété suivante par récurrence sur l de 1 à ℓ :

$$V_l = \prod_{\substack{(G,m)\in \mathrm{fsc}(F)\\ m\geq l}} G, \qquad W_l = \sum_{\substack{(G,m)\in \mathrm{fsc}(F)\\ m\geq l}} (m-l+1)\frac{V_l}{G}G'.$$

À l'étape (2) nous avons :

$$U = \mathrm{pgcd}(F,F'), \qquad V = \frac{F}{\mathrm{pgcd}(F,F')}, \qquad W = \frac{F'}{\mathrm{pgcd}(F,F')}.$$

Comme nous avons vu précedemment que $\mathrm{pgcd}(F,F') = \prod_{(G,m)\in \mathrm{fsc}(F)} G^{m-1}$, l'hypothèse de récurrence est satisfaite pour $l = 1$. Supposons l'hypothèse de récurrence satisfaite pour une certaine valeur de l dans $\{1,\dots,\ell-1\}$. En utilisant le Lemme 18.1 nous déduisons :

$$H_l = \mathrm{pgcd}(V_l, W_l - V_l') = \mathrm{pgcd}\left(\prod_{\substack{(G,m)\in \mathrm{fsc}(F)\\ m\geq l}} G, \sum_{\substack{(G,m)\in \mathrm{fsc}(F)\\ m\geq l}} (m-l)\frac{V_l}{G}G'\right) = \prod_{\substack{(G,m)\in \mathrm{fsc}(F)\\ m= l}} G,$$

ce qui conduit aux formules attendues pour V_{l+1} et W_{l+1}. Puisque ℓ est le premier entier tel que $\deg(V_\ell) = 0$, nous obtenons que $\ell - 1$ est la plus grande des multiplicités des facteurs de F. Finalement L contient tous les facteurs sans carré de F à la fin de la boucle.

L'étape (2) utilise $O(M(n)\log n)$ opérations dans $\mathbb{K}$. L'étape l de la boucle coûte au plus $O(M(\deg(V_l))\log(\deg(V_l)))$ puisque $\deg(W_l) \leq \deg(V_l) - 1$. Par la super-additivité de M, l'étape (3) totalise au plus $O\left(M\left(\sum_{l=1}^{\ell-1}\deg(V_l)\right)\log n\right)$ opérations dans $\mathbb{K}$, ce qui conduit à la borne de complexité attendue puisque $\prod_{l=1}^{\ell-1} V_l = F$. ■

18.3 Séparabilité et déflation

Rappelons que le corps fini à q éléments est noté $\mathbb{F}_q$. Si p est premier alors $\mathbb{F}_p$ n'est autre que $\mathbb{Z}/p\mathbb{Z}$. Les rappels mathématiques sur les corps finis sont regroupés au début du chapitre suivant.

Si $\mathbb{K}$ est le corps des fractions $\mathbb{F}_2(T)$, et si $F(X) = T^2 + X^2 = (T+X)^2$, alors $F'(X) = 0$, et ni l'algorithme de Musser ni celui de Yun ne calcule la factorisation sans carré de F. Nous allons voir que le « bon concept » en caractéristique non nulle n'est pas celui d'être sans carré, mais plutôt d'être séparable :

Définition 18.1 Un polynôme $F \in \mathbb{K}[X]$ est dit *séparable* s'il n'a aucune racine multiple dans une clôture algébrique $\bar{\mathbb{K}}$ de $\mathbb{K}$.

Nous rappelons la définition du discriminant, déjà abordée dans l'Exemple 6.7 page 118.

Définition 18.2 Le *discriminant* de $F \in \mathbb{K}[X]$ de degré n, noté $\mathrm{disc}(F)$, est défini par

$$\mathrm{Res}(F, F') = (-1)^{n(n-1)/2}\,\mathrm{ct}(F)\,\mathrm{disc}(F),$$

où $\mathrm{ct}(F)$ représente le coefficient de degré n de F.

Proposition 18.5 Un polynôme F qui n'est pas constant est séparable si, et seulement si, son discriminant est non nul.

Démonstration. Notons $\alpha_1, \ldots, \alpha_r$ les racines distinctes de F dans $\bar{\mathbb{K}}$ de multiplicités respectives $m_1, \ldots, m_r$. On peut supposer F unitaire. Si toutes les multiplicités valent 1, alors $F'(\alpha_i) = \prod_{j \neq i}(\alpha_i - \alpha_j) \neq 0$. Par la formule de Poisson (Théorème 6.5), le discriminant est égal au produit des $F'(\alpha_i)$, il est donc non nul. Si l'une des multiplicités, disons $m_1 \geq 2$ alors $F'(\alpha_1) = 0$ et le discriminant s'annule. ■

Par exemple, si $\mathbb{K}$ est le corps des fractions $\mathbb{F}_2(T)$, alors $F(X) = T + X^2$ est sans carré mais n'est pas séparable pour autant puisque $F'(X)$ est identiquement nul. Néanmoins si $p = 0$, alors la condition de séparabilité est équivalente à être sans carré.

Dans la suite nous noterons $\mathcal{B} = \{1, p, p^2, p^3, \ldots\}$ l'ensemble des puissances de p si $p > 0$, et $\mathcal{B} = \{1\}$ si $p = 0$. Les problèmes liés à la séparabilité concernent les cas de caractéristique non nulle. Nous aurons besoin des définitions et propositions suivantes.

Définition 18.3 Le *degré d'inséparabilité* de F, noté $\deg^i(F)$, est le plus grand élément q de $\mathcal{B}$ tel que $F \in \mathbb{K}[X^q] \setminus \mathbb{K}[X^{pq}]$.

Proposition 18.6 Soient F et G deux polynômes de $\mathbb{K}[X] \setminus \mathbb{K}$.

1. Le degré d'inséparabilité de F est la plus grande puissance de p qui divise la multiplicité de chaque racine de F.
2. $\deg^i(FG) \geq \min(\deg^i(F), \deg^i(G))$, l'inégalité devient une égalité dès lors que F et G sont premiers entre eux.

Démonstration. En utilisant $(X - a)^q = X^q - a^q$ pour tout a dans une clôture algébrique de $\mathbb{K}$, si $q \in \mathcal{B}$ divise la multiplicité de chaque racine de F, alors F appartient bien à $\mathbb{K}[X^q]$. Réciproquement, si F peut s'écrire $G(X^q)$ avec $q \in \mathcal{B}$, alors la multiplicité d'une racine de F est bien un multiple de q. La partie (2) est une conséquence directe de la partie (1). ■

Définition 18.4 Le *degré de séparabilité* de $F \in \mathbb{K}[X]$, noté $\deg^s(F)$, est défini comme

suit :

$$\deg^s(F) = \sum_{(G,m)\in \mathrm{firr}(F)} \deg(G)/\deg^i(G) = \sum_{(G,m)\in \mathrm{firr}(F)} \deg^s(G).$$

Exemple 18.1 Soient $\mathbb{K} = \mathbb{F}_2(T)$ et $F(X) = X^4 + X^2 + 1 + T$. On a $\deg^i(F) = 2$, et comme F est irréductible, $\deg^s(F) = 2$.

Proposition 18.7 Soient F et G des polynômes de $\mathbb{K}[X] \setminus \mathbb{K}$.
1. $\deg^s(F)$ est égal au nombre de racines de F dans $\bar{\mathbb{K}}$ sans compter les multiplicités.
2. F est séparable si, et seulement si, $\deg^s(F) = \deg(F)$.
3. Pour tout $q \in \mathcal{B}$, $\deg^s(F(X^q)) = \deg^s(F)$.
4. $\deg^s(FG) \leq \deg^s(F) + \deg^s(G)$, avec égalité si, et seulement si, F et G sont premiers entre eux.

Démonstration. Ces propriétés découlent presque immédiatement des définitions et nous laissons au lecteur le soin de les vérifier à titre d'exercice. ■

Lorsque qu'un polynôme F appartient à $\mathbb{K}[X^p]$, il est utile de le voir comme un polynôme en la variable X^p. Ceci motive la définition suivante :

Définition 18.5 Si $F \in \mathbb{K}[X] \setminus \mathbb{K}$, le *polynôme déflaté* de F est l'unique polynôme $\tilde{F}$ défini par

$$\tilde{F}(X^{\deg^i(F)}) = F(X).$$

Les multiplicités des racines de $\tilde{F}$ sont celles de F divisées par $\deg^i(F)$, d'où la terminologie « déflation ». Puisque $\tilde{F}$ appartient à $\mathbb{K}[X] \setminus \mathbb{K}[X^p]$, la fonction suivante Φ de déflation est bien définie :

$$\begin{aligned} \Phi : \mathbb{K}[X] \setminus \mathbb{K} &\to (\mathbb{K}[X] \setminus \mathbb{K}[X^p]) \times \mathcal{B} \\ F &\mapsto (\tilde{F}, \deg^i(F)). \end{aligned}$$

Exemple 18.2 Avec l'exemple précédent où $\mathbb{K} = \mathbb{F}_2(T)$ et $F(X) = X^4 + X^2 + 1 + T$, on a $\Phi(F) = (X^2 + X + 1 + T, 2)$.

Notons $\mathcal{Q}$ l'ensemble des puissances q-ièmes des polynômes irréductibles de $\mathbb{K}[X] \setminus \mathbb{K}$ pour $q \in \mathcal{B}$, et $\mathcal{S}$ le sous-ensemble des polynômes séparables irréductibles.

Proposition 18.8 Φ est une bijection qui envoie $\mathcal{Q}$ surjectivement sur $\mathcal{S} \times \mathcal{B}$.

Démonstration. Le fait que Φ est une bijection découle clairement des définitions. Si $F \in \mathbb{K}[X]$ est irréductible, alors $\tilde{F}$ est nécessairement irréductible et donc séparable

car $\tilde{F} \notin \mathbb{K}[X^p]$. Puisque le polynôme déflaté de F^q coincide avec celui de $\tilde{F}^q$ pour tout $q \in \mathcal{B}$, le Lemme 18.9 ci-dessous implique que $\Phi(\mathcal{Q})$ appartient bien à $\mathcal{S} \times \mathcal{B}$.

Réciproquement, si $(G, q) \in \mathcal{S} \times \mathcal{B}$. Nous devons montrer que $F(X) = G(X^q)$ appartient à $\mathcal{Q}$, et que $\deg^i(F) = q$. Cette dernière égalité est claire puisque G est séparable. Soit $h \leq q$ le plus grand élément de $\mathcal{B}$ tel que $G(X^h)$ est une puissance h-ième, et soit $\tilde{G}$ la racine h-ième correspondante, de sorte que $\tilde{G}^h(X) = G(X^h)$. Par le Lemme 18.9 ci-dessous, $\tilde{G}$ et donc $\tilde{G}(X^{q/h})$ sont irréductibles et séparables, ce qui prouve que $F(X) = \tilde{G}(X^{q/h})^h \in \mathcal{Q}$. ■

Lemme 18.9 Soient F et G deux polynômes de $\mathbb{K}[X] \setminus \mathbb{K}$ tels que $G(X^p) = F(X)^p$.
1. G est séparable si, et seulement si, F est séparable.
2. Si G est irréductible alors F est irréductible. La réciproque est vraie si F ou G est séparable.

Démonstration. La preuve de la partie (a) est immédiate. Les multiplicités de $G(X^p)$ (resp. de $F(X)^p$) sont toutes p si, et seulement si, G (resp. F) est séparable. La partie (b) est donc une conséquence de la Proposition 18.7. Concernant la partie (c), si G est irréductible, alors pour n'importe quel diviseur strict A de F, le polynôme A^p divise F^p, et donc B divise G, si nous définissons B par $B(X^p) = A(X)^p$. Par conséquent B et donc A sont dans $\mathbb{K}$. Réciproquement, supposons F irréductible, et soit A un diviseur strict de G. Puisque $A(X^p)$ divise $F(X)^p$, le polynôme $A(X^p)$ peut s'écrire $F(X)^\alpha$ pour un certain $\alpha \in \{0, \dots, p-1\}$. En dérivant, nous obtenons $\alpha F' F^{\alpha-1} = 0$, d'où $\alpha = 0$ dès que F est séparable. ■

18.4 Factorisation séparable

Nous pouvons maintenant définir un nouveau type de factorisation correspondant à écrire F essentiellement comme un produit de facteurs séparables :

Définition 18.6 La *factorisation séparable* d'un polynôme $F \in \mathbb{K}[X] \setminus \mathbb{K}$, notée fsep(F), est l'ensemble des triplets $(G_1, q_1, m_1), \dots, (G_s, q_s, m_s)$ de $\mathbb{K}[X] \times \mathcal{B} \times \mathbb{N}^*$ vérifiant les propriétés suivantes :
- (S_1) $F(X) = \prod_{i=1}^{s} G_i^{m_i}(X^{q_i})$,
- (S_2) les $G_i(X^{q_i})$ sont premiers entre eux deux à deux,
- (S_3) les m_i ne sont pas divisibles par p,
- (S_4) les G_i sont séparables de degré au moins 1,
- (S_5) les couples (q_i, m_i) sont deux à deux distincts.

Notons que dans le cas où $p = 0$ la factorisation séparable n'est rien d'autre que la factorisation sans carré.

Théorème 18.10 Tout polynôme F de $\mathbb{K}[X]$ admet une unique factorisation séparable (à permutation et unités près dans $\mathbb{K}$).

Démonstration. Soient $(F_1, e_1), \ldots, (F_r, e_r)$ la factorisation irréductible de F dans $\mathbb{K}[X]$. Pour chaque $i \in \{1, \ldots, r\}$, soit a_i la plus grande puissance de p qui divise e_i, et soit $m_i = e_i/a_i$. En définissant $(G_i, q_i) = \Phi(F_i^{a_i})$, nous obtenons des triplets

$$(G_1, q_1, m_1), \ldots, (G_s, q_s, m_s)$$

qui satisfont (S_1), (S_2) et (S_3). La propriété (S_4) est aussi satisfaite par la Proposition 18.8.

Pour obtenir (S_5) il suffit de regrouper les triplets qui partagent les mêmes deuxièmes et troisièmes coordonnées comme suit : si $(A_1, q, m), \ldots, (A_t, q, m)$ sont de tels triplets alors nous les fusionnons en le triplet $(A_1 \cdots A_r, q, m)$. Les propriétés de (S_1) à (S_4) sont préservées, ce qui prouve l'existence de la factorisation séparable.

Supposons maintenant que nous disposons de triplets

$$(G_1, q_1, m_1), \ldots, (G_s, q_s, m_s)$$

qui satisfont les propriétés de (S_1) à (S_5). Pour chaque $i \in \{1, \ldots, s\}$, n'importe quelle racine de $G_i(X^{q_i})$ dans $\bar{\mathbb{K}}$ admet q_i comme multiplicité, et donc $q_i m_i$ comme multiplicité dans F. La propriété (S_5) implique alors que les racines de $G_i(X^{q_i})$ sont exactement celles de F de multiplicité $q_i m_i$, ce qui conduit à l'unicité de la factorisation séparable. ■

Exemple 18.3 Lorsque $\mathbb{K} = \mathbb{F}_3$ et $F = X^2(X+1)^3(X+2)^4 = X^9 + 2X^8 + 2X^3 + X^2$, nous avons $\mathrm{fsep}(F) = \{(X, 1, 2), (X+1, 3, 1), (X+2, 1, 4)\}$.

Exemple 18.4 Lorsque $\mathbb{K} = \mathbb{F}_3(T)$ et $F = (X + 2T)^7(X^3 + 2T)^3(X^6 + T)$, nous avons $\mathrm{fsep}(F) = \{(X + 2T, 1, 7), (X + 2T^3, 9, 1), (X^2 + T, 3, 1)\}$.

Algorithme de Gianni et Trager

Les algorithmes de factorisation séparable connus utilisent ceux de Musser ou de Yun vus précédemment. Dans le cas de l'algorithme de Musser, le point de départ est la propriété suivante :

Proposition 18.11 Si l'Algorithme 18.1 est exécuté en caractéristique non nulle, et si $L = \{(G_1, m_1), \ldots, (G_s, m_s)\}$ est la liste renvoyée, alors

$$(G_1, 1, m_1), \ldots, (G_s, 1, m_s)$$

sont exactement les facteurs séparables de F de la forme $(G, 1, m)$.

Démonstration. Notons F_0 le facteur de F composé des racines de multiplicité qui ne sont pas multiples de p et posons $F_1 = F/F_0$. Notons justement $(G_1, 1, m_1), \ldots, (G_s, 1, m_s)$ la factorisation séparable de F_0. Nous allons montrer que l'algorithme renvoie bien $\{(G_1, m_1), \ldots, (G_s, m_s)\}$.

Entrée Un polynôme $F \in \mathbb{K}[X]$ de degré $n \geq 1$.
Sortie La factorisation séparable de F.
1. Appeler l'Algorithme 18.1 pour obtenir l'ensemble $L = \{(G_1, 1, m_1), \ldots, (G_r, 1, m_r)\}$ des facteurs séparables de F contenant toutes les racines de F de multiplicité non divisible par p.
2. Calculer $C = F/(G_1^{m_1} \cdots G_s^{m_s})$ et construire $\tilde{C}$ tel que $\tilde{C}(X^p) = C(X)$.
3. Appeler récursivement le présent algorithme pour obtenir la factorisation séparable
   $\tilde{L} = \{(G_{r+1}, q_{r+1}, m_{r+1}), \ldots, (G_s, q_s, m_s)\}$ de $\tilde{C}$.
4. Renvoyer $\{(G_1, 1, m_1), \ldots, (G_r, 1, m_r)\}$
   $\cup \{(G_{r+1}, pq_{r+1}, m_{r+1}), \ldots, (G_s, pq_s, m_s)\}$.

Algorithme 18.3 – Algorithme de Gianni et Trager.

Comme $F_1' = 0$ nous obtenons

$$F' = F_1 \sum_{i=1}^{s} m_i G_i' G_i^{m_i - 1} \frac{F_0}{G_i^{m_i}},$$

alors, à l'étape (2) de l'algorithme, nous avons $C = F_1 \prod_{i=1}^{s} G_i^{m_i - 1}$, et $G = \prod_{i=1}^{s} G_i$. Dans la première étape de la boucle nous obtenons $P = \prod_{i=1, m_i \geq 2}^{s} G_i$, $C = F_1 \prod_{i=1, m_i \geq 2}^{s} G_i^{m_i - 2}$, et G/P correspond aux G_i pour lesquels $m_i = 1$. À la deuxième étape de la boucle nous avons

$$P = \prod_{i=1, m_i \geq 3}^{s} G_i, \quad C = F_1 \prod_{i=1, m_i \geq 3}^{s} G_i^{m_i - 3},$$

et G/P correspond aux G_i pour lesquels $m_i = 2$. La preuve s'achève ainsi par récurrence comme pour l'algorithme de Musser. ■

En conservant les notations de la preuve, notons F_0 le facteur de F composé des racines de multiplicité qui ne sont pas multiples de p et posons $F_1 = F/F_0$. En introduisant $\tilde{F}_1(X^p) = F_1(X)$, le degré de $\tilde{F}_1$ est au plus celui de F divisé par p, et nous pouvons donc calculer récursivement la factorisation séparable de $\tilde{F}_1$. Cette approche conduit à un algorithme dû à Gianni et Trager, présenté dans l'Algorithme 18.3.

Proposition 18.12 L'Algorithme 18.3 est correct. Si $\mathbb{K}$ est un corps, alors son coût est borné par $O(n\mathsf{M}(n)\log n)$.

Démonstration. Par construction, les racines de C ont une multiplicité divisible par p donc $\tilde{C}$ est bien défini à l'étape (2). Comme $\tilde{L}$ est la factorisation séparable de $\tilde{C}$, alors $\{(G_{r+1}, pq_{r+1}, m_{r+1}), \ldots, (G_s, pq_s, m_s)\}$ est la factorisation séparable de C. Comme

le degré de $\tilde{C}$ est borné par n/p, et que le calcul de C demande $O(M(n)\log n)$ opérations dans $\mathbb{K}$ par la Proposition 5.5, le coût total est borné par

$$O\left(\sum_{i\geq 1}\frac{n}{p^i}M\left(\frac{n}{p^i}\right)\log\frac{n}{p^i}\right)\subseteq O(nM(n)\log n). \qquad \blacksquare$$

Algorithme de Lecerf

L'extension de l'algorithme de Yun pour effectuer la factorisation séparable conduit à un coût arithmétique quasi-linéaire. Cette fin de section est assez technique et peut être omise en première lecture. Le point de départ est le lemme suivant :

Lemme 18.13 Supposons $p > 0$, et soit F un polynôme unitaire de $\mathbb{K}[X]$. Il existe des uniques polynômes unitaires S_0 et S_1 de $\mathbb{K}[X]$ avec les propriétés suivantes :
- les facteurs irréductibles de S_0 sont séparables de multiplicité au plus $p-1$,
- $F(X) = S_0(X)S_1(X^p)$.

Démonstration. Le polynôme S_0 est défini par sa factorisation en irréductibles :

$$\mathrm{firr}(S_0) = \{(G, m \bmod p) \mid (G,m)\in \mathrm{firr}(F), G \text{ est séparable et } m \bmod p \neq 0\}.$$

Un facteur irréductible G de F/S_0 est soit inséparable soit de multiplicité m dans F divisible par p. Par conséquent $F/S_0 \in \mathbb{K}[X^p]$ et nous définissons $S_1(X^p) = F(X)/S_0(X)$. $\blacksquare$

Exemple 18.5 Avec l'Exemple 18.3 nous avons $S_0 = (X+2)X^2$ et $S_1 = (X+1)(X+2)$.

Exemple 18.6 Avec l'Exemple 18.4 nous avons $S_0 = X + 2T$ et $S_1 = (X^2+T)(X+2T^3)^2(X+2T)^3$.

Pour calculer la factorisation séparable de F nous commençons par celle de S_0.

Lemme 18.14 Si l'Algorithme 18.2 est utilisé en caractéristique non nulle p, alors il renvoie la factorisation sans carré de S_0 tel que défini dans le Lemme 18.13. De plus sa complexité arithmétique reste en $O(M(n)\log n)$.

Démonstration. À l'étape (2) de l'algorithme nous avons :

$$U = \mathrm{pgcd}(S_0, S_0')S_1(X^p), \qquad V = \frac{S_0}{\mathrm{pgcd}(S_0,S_0')}, \qquad W = \frac{S_0'}{\mathrm{pgcd}(S_0,S_0')}.$$

Nous avons les mêmes valeurs de V et W en appelant l'algorithme directement avec S_0. Ensuite, nous laissons le soin au lecteur de vérifier à titre d'exercice que l'algorithme de Yun renvoie bien la factorisation sans carré de S_0, qui coïncide par ailleurs avec la factorisation séparable. Finalement l'analyse de complexité reste inchangée. $\blacksquare$

Entrée $F \in \mathbb{K}[X]$ de degré n, $\mathrm{fsc}(S_0)$ et $\mathrm{fsep}(S_1)$.
Sortie $\mathrm{fsep}(F)$.

1. Si $S_1 \in \mathbb{K}$ alors renvoyer $\{(G_1,1,m_1),\dots,(G_r,1,m_r)\}$.
   Sinon initialiser L avec la liste vide.
2. Pour tout $i \in \{1,\dots,r\}$ faire :
   a. pour tout $j \in \{1,\dots,s\}$, calculer
      $R_{i,j}(X) = \tilde{G}_i(X) \bmod H_j(X^{q_j})$,
      où $\tilde{G}_i$ est le polynôme défini par $\tilde{G}(X^p) = G_i(X)^p$;
   b. pour tout $j \in \{1,\dots,s\}$, calculer
      $S_{i,j}(X) = \mathrm{pgcd}(R_{i,j}(X), H_j(X^{q_j}))$;
   c. pour tout $j \in \{1,\dots,s\}$, calculer
      $T_{i,j}(X) = G_i(X) \bmod S_{i,j}(X^p)$;
   d. pour tout $j \in \{1,\dots,s\}$, calculer
      $U_{i,j}(X) = \mathrm{pgcd}(T_{i,j}(X), S_{i,j}(X^p))$
      et adjoindre $(U_{i,j}, 1, m_i + pq_jn_j)$ à L si $\deg(U_{i,j}) \geq 1$;
   e. calculer $V_i(X) = \prod_{j=1}^{s} U_{i,j}(X)$;
   f. calculer $U_{i,0}(X) = G_i(X)/V_i(X)$
      et adjoindre $(U_{i,0}, 1, m_i)$ à L si $\deg(U_{i,0}) \geq 1$.
3. Pour tout $j \in \{1,\dots,s\}$ faire :
   a. calculer $W_j(X^{q_j}) = \prod_{i=1}^{r} S_{i,j}(X)^{q_j}$;
   b. calculer $U_{0,j}(X) = H_j(X)/W_j(X)$
      et adjoindre $(U_{0,j}, pq_j, n_j)$ à L si $\deg(U_{0,j}) \geq 1$.
4. Renvoyer L.

Algorithme 18.4 – Algorithme rapide de fusion des factorisations séparables.

Pour obtenir la factorisation séparable de F, nous commençons par calculer $\mathrm{fsc}(S_0)$ et S_1 par l'algorithme précédent. Ensuite nous calculons récursivement $\mathrm{fsep}(S_1)$, pour finalement fusionner ces factorisations et obtenir $\mathrm{fsep}(F)$. Expliquons maintenant comment cette fusion peut être réalisée rapidement. Si

$$\mathrm{fsc}(S_0) = \{(G_1,m_1),\dots,(G_r,m_r)\}, \quad \mathrm{fsep}(S_1) = \{(H_1,q_1,n_1),\dots,(H_s,q_s,n_s)\},$$

alors nous commençons par calculer tous les $U_{i,j} = \mathrm{pgcd}(G_i, H_j(X^{pq_j}))$ pour $i \in \{1,\dots,r\}$ et $j \in \{1,\dots,s\}$. Nous verrons que $(U_{i,j}, 1, m_i + pq_jn_j)$ est un élément de $\mathrm{fsep}(F)$ dès que $\deg(U_{i,j}) \geq 1$.

Lemme 18.15 L'Algorithme 18.4 est correct et utilise $O(M(n)\log n)$ opérations arithmétiques dans $\mathbb{K}$.

Démonstration. Nous devons vérifier que la liste L renvoyée par l'algorithme satisfait les propriétés (S_1) à (S_5) de la Définition 18.6. Pour tout $i \in \{1,\dots,r\}$ et tout $j \in \{1,\dots,s\}$ nous avons

$$R_{i,j}(X^p) = G_i(X)^p \bmod H_j(X^{pq_j}), \quad S_{i,j}(X^p) = \mathrm{pgcd}(G_i(X)^p, H_j(X^{pq_j})),$$

et par conséquent

$$\begin{aligned}\mathrm{U}_{i,j}(\mathrm{X}) &= \operatorname{pgcd}(\mathrm{G}_i(\mathrm{X}), \mathrm{S}_{i,j}(\mathrm{X}^p)) = \operatorname{pgcd}(\mathrm{G}_i(\mathrm{X}), \operatorname{pgcd}(\mathrm{G}_i(\mathrm{X})^p, \mathrm{H}_j(\mathrm{X}^{pq_j})))\\ &= \operatorname{pgcd}(\mathrm{G}_i(\mathrm{X}), \mathrm{H}_j(\mathrm{X}^{pq_j})).\end{aligned}$$

À la fin de l'étape (2), pour tout $i \in \{1,\dots,r\}$, il est clair que $\mathrm{G}_i = \prod_{j=0}^{s} \mathrm{U}_{i,j}$, et que les $\mathrm{U}_{i,j}$ sont séparables et premiers entre eux. Dans l'étape (3) nous avons

$$\mathrm{S}_{i,j}(\mathrm{X})^{q_j} = \operatorname{pgcd}(\tilde{\mathrm{G}}_i(\mathrm{X})^{q_j}, \mathrm{H}_j(\mathrm{X}^{q_j})^{q_j}) = \operatorname{pgcd}(\tilde{\mathrm{G}}_i(\mathrm{X})^{q_j}, \mathrm{H}_j(\mathrm{X}^{q_j}))$$

puisque les $\tilde{\mathrm{G}}_i$ sont séparables par le Lemme 18.9. Par conséquent les $\mathrm{U}_{0,j}$ sont bien définis et nous avons

$$\mathrm{W}_j(\mathrm{X}^{pq_j}) = \prod_{i=1}^{r} \mathrm{S}_{i,j}(\mathrm{X}^p)^{q_j} = \prod_{i=1}^{r} \operatorname{pgcd}(\mathrm{G}_i(\mathrm{X})^{pq_j}, \mathrm{H}_j(\mathrm{X}^{pq_j})) = \prod_{i=1}^{r} \mathrm{U}_{i,j}(\mathrm{X})^{pq_j}.$$

Nous en déduisons que $\mathrm{H}_j(\mathrm{X}^{pq_j}) = \mathrm{U}_{0,j}(\mathrm{X}^{pq_j}) \prod_{i=1}^{r} \mathrm{U}_{i,j}(\mathrm{X})^{pq_j}$ est vrai pour tout $j \in \{1,\dots,s\}$. Nous pouvons récrire F en :

$$\begin{aligned}\mathrm{F}(\mathrm{X}) = \mathrm{S}_0(\mathrm{X})\mathrm{S}_1(\mathrm{X}^p) &= \left(\prod_{i=1}^{r}\prod_{j=0}^{s} \mathrm{U}_{i,j}(\mathrm{X})^{m_i}\right) \prod_{j=1}^{s} \left(\mathrm{U}_{0,j}(\mathrm{X}^{pq_j})^{n_j} \prod_{i=1}^{r} \mathrm{U}_{i,j}(\mathrm{X})^{pq_j n_j}\right)\\ &= \left(\prod_{i=1}^{r}\prod_{j=1}^{s} \mathrm{U}_{i,j}(\mathrm{X})^{m_i + pq_j n_j}\right)\left(\prod_{i=1}^{r} \mathrm{U}_{i,0}(\mathrm{X})^{m_i}\right)\left(\prod_{j=1}^{s} \mathrm{U}_{0,j}(\mathrm{X}^{pq_j})^{n_j}\right).\end{aligned}$$

La factorisation renvoyée dans L satisfait bien (S_1). Les autres propriétés sont immédiatement satisfaites par construction.

Si $p \geq n+1$ alors $\mathrm{S}_1 \in \mathbb{K}$, et l'algorithme ne fait rien. Nous pouvons donc supposer à partir de maintenant que $p \leq n$. Le coût suivant est une conséquence directe de la super-additivité de M, de l'inégalité $r \leq p-1$, et de $\deg(\mathrm{S}_0) + p\deg(\mathrm{S}_1) = n$. L'étape (2.a) calcule des puissances p-ièmes et des réductions simultanées, totalisant ainsi

$$\begin{aligned}&\mathrm{O}\left(\sum_{i=1}^{r} \deg(\mathrm{G}_i)\log p + \sum_{i=1}^{r} (\mathsf{M}(\deg(\mathrm{S}_1))\log s + \mathsf{M}(\max(\deg(\mathrm{G}_i), \deg(\mathrm{S}_1))))\right)\\ &\qquad \subseteq \mathrm{O}(\deg(\mathrm{S}_0)\log d + p\mathsf{M}(\mathrm{S}_1)\log d + \mathsf{M}(\deg(\mathrm{S}_0))) \subseteq \mathrm{O}(\mathsf{M}(n)\log n).\end{aligned}$$

Le coût total des pgcd dans les étapes (2.b) tombe dans

$$\mathrm{O}\left(\sum_{i=1}^{r}\sum_{j=1}^{s} \mathsf{M}(\deg(\mathrm{H}_j(\mathrm{X}^{q_j})))\log(\deg(\mathrm{H}_j(\mathrm{X}^{q_j})))\right) \subseteq \mathrm{O}(p\mathsf{M}(\deg(\mathrm{S}_1))\log n) \subseteq \mathrm{O}(\mathsf{M}(n)\log n).$$

Le coût total des réductions simultanées des étapes (2.c) tombe dans

$$\begin{aligned}&\mathrm{O}\left(\sum_{i=1}^{r}\left(\mathsf{M}(\deg(\mathrm{G}_i)) + \mathsf{M}\left(p\sum_{j=1}^{s}\deg(\mathrm{S}_{i,j})\right)\log n\right)\right)\\ &\qquad \subseteq \mathrm{O}(\mathsf{M}(\deg(\mathrm{S}_0)) + \mathsf{M}(p\deg(\mathrm{S}_1))\log n) \subseteq \mathrm{O}(\mathsf{M}(n)\log n).\end{aligned}$$

Le coût total des étapes (2.d) est borné par

$$\mathrm{O}\left(\sum_{i=1}^{r}\sum_{j=1}^{s}\mathsf{M}(p\deg(\mathrm{S}_{i,j}))\log n\right)\subseteq \mathrm{O}(\mathsf{M}(n)\log n).$$

Les étapes (2.e) et (2.f) coûtent

$$\mathrm{O}\left(\sum_{i=1}^{r}\mathsf{M}(\deg(\mathrm{G}_i))\log n\right)\subseteq \mathrm{O}(\mathsf{M}(n)\log n).$$

Finalement le coût de l'étape (3) est au plus

$$\mathrm{O}\left(\sum_{j=1}^{s}\mathsf{M}(\deg(\mathrm{H}_j(\mathrm{X}^{q_j})))\log n\right)\subseteq \mathrm{O}(\mathsf{M}(n)\log n).$$ ■

Exemple 18.7 Avec l'Exemple 18.3 nous entrons dans l'Algorithme 18.4 avec

$$\mathrm{fsc}(\mathrm{S}_0)=\{(\mathrm{X}+2,1),(\mathrm{X},2)\},\quad \mathrm{fsep}(\mathrm{S}_1)=\{((\mathrm{X}+1)(\mathrm{X}+2),1,1)\},$$

et obtenons les valeurs suivantes pour les $\mathrm{U}_{i,j}$:

$i\setminus j$	0	1
0		$\mathrm{X}+1$
1	1	$\mathrm{X}+2$
2	X	1

Exemple 18.8 Avec l'Exemple 18.4 nous entrons dans l'Algorithme 18.4 avec

$$\mathrm{fsc}(\mathrm{S}_0)=\{(\mathrm{X}+2\mathrm{X},1)\},\quad \mathrm{fsc}(\mathrm{S}_1)=\{(\mathrm{X}^2+\mathrm{T},1,1),(\mathrm{X}+2\mathrm{T}^3,1,2),(\mathrm{X}+2\mathrm{T}^3,3,1)\},$$

et obtenons les valeurs suivantes pour les $\mathrm{U}_{i,j}$:

$i\setminus j$	0	1	2	3
0		$\mathrm{X}^2+\mathrm{T}$	1	$\mathrm{X}+2\mathrm{T}^3$
1	1	1	$\mathrm{X}+2\mathrm{T}$	1

Finalement les résultats de cette section se résument comme suit :

Théorème 18.16 La factorisation séparable d'un polynôme $\mathrm{F}\in\mathbb{K}[\mathrm{X}]$ de degré n peut être calculée au moyen de $\mathrm{O}(\mathsf{M}(n)\log n)$ opérations dans $\mathbb{K}$.

Démonstration. La preuve est une conséquence directe des Lemmes 18.14 et 18.15, et de la borne $\sum_{k\geq 0}\mathsf{M}(n/p^k)\log(n/p^k)\in\mathrm{O}(\mathsf{M}(n)\log n)$. ■

Entrée $(G, q) \in \mathcal{S} \times \mathcal{B}$.
Sortie $(H, h) \in \mathbb{K}[X] \times \mathcal{B}$ avec H irréductible tel que $H^h = \Phi^{-1}(G, q)$.
1. Poser $\tilde{G} = G$ et $h = 1$.
2. Tant que $\tilde{G}(X^p)$ est une puissance p-ième et tant que $h < q$, remplacer $\tilde{G}$ par la racine p-ième de $\tilde{G}(X^p)$, et multiplier h par p.
3. Renvoyer $(\tilde{G}(X^{q/h}), h)$.

Algorithme 18.5 – Calcul de l'inverse de la fonction de déflation.

Entrée Un polynôme $F \in \mathbb{K}[]$ de degré $n \geq 1$.
Sortie firr(F).
1. Calculer la factorisation séparable fsep(F) de F.
2. Pour chaque $(G, q, m) \in$ fsep(F) calculer la factorisation irréductible de G.
3. Renvoyer

$$\bigcup_{(G,q,m)\in \mathrm{fsep}(F)} \{(H(X^{q/h}), hm) \mid H^h = \Phi^{-1}(\bar{G}, q), \text{ pour chaque facteur irréductible } \bar{G} \text{ de } G\}.$$

Algorithme 18.6 – Réduction de la factorisation au cas séparable.

18.5 Réduction à la factorisation des polynômes séparables

Une fois la factorisation séparable effectuée nous verrons dans les chapitres suivants comme réaliser la factorisation irréductible de chacun des facteurs séparables dans les situations courantes. Nous finissons donc ce chapitre en expliquant comment le problème de la factorisation irréductible se réduit au cas des polynômes séparables.

Nous supposons que $\mathbb{K}$ dispose d'une opération élémentaire permettant d'extraire une racine p-ième, et expliquons, dans l'Algorithme 18.5, comment implémenter l'inverse de la fonction de déflation Φ^{-1}.

Lemme 18.17 L'Algorithme 18.5 est correct et coûte $O(\deg(G) \log_p q)$ extractions de racines p-ièmes de $\mathbb{K}$.

Démonstration. Les quantités $\tilde{G}$ est h calculées par l'algorithme coïncident avec celles de la preuve de la Proposition 18.8. ■

Dans l'Algorithme 18.6 nous supposons disposer d'un algorithme de factorisation irréductible pour les polynômes séparables.

Proposition 18.18 L'Algorithme 18.6 est correct. Il effectue des factorisations irreducibles de polynômes séparables de $\mathbb{K}[X]$ dont la somme des degrés est au

plus n, ainsi que $O(M(n)\log n)$ opérations arithmétiques dans $\mathbb{K}$ et $O(n)$ extractions de racines p-ièmes dans $\mathbb{K}$.

Démonstration. Le coût de la première étape vient du Théorème 18.16, et le coût de la dernière étape du lemme précédent. ■

Notes

Cette partie sur la factorisation des polynômes est adaptée des notes d'un cours donné à l'occasion des *Journées Nationales de Calcul Formel* en 2013 [Lec13].

La recherche des racines d'un polynôme à une variable et plus généralement la factorisation de tels polynômes en irréductibles trouve ses origines dans des temps très anciens des mathématiques. Un bref historique a été donné par von zur Gathen [Gat06]. Une présentation plus détaillée sur la factorisation des polynômes se trouve dans les sections « Notes » des chapitres de la partie III de son livre avec Gerhard [GG03]. Nous fournirons des éclairages historiques plus précis dans les chapitres suivants. Précisons néanmoins dès maintenant que la plupart des logiciels de calcul formel offrent des fonctions pour factoriser les entiers et les polynômes. Le logiciel Magma propose une implémentation efficace couvrant tous les cas habituels [Ste05]. Parmi les logiciels libres, de nombreux cas particuliers de factorisation sont disponibles dans NTL, Pari/gp, Sage, Singular, et plus récemment dans Mathemagix.

Concernant la factorisation sans carré, l'algorithme original de Musser remonte à 1971 [Mus71]. Son extension à la factorisation séparable est due à Gianni et Trager [GT96].

L'algorithme de Yun [Yun76] a été étendu à la factorisation séparable par Lecerf [Lec08]. Par ailleurs, lorsque $\mathbb{K}$ est parfait, cette extension coïncide essentiellement avec l'algorithme de factorisation sans carré proposé par Knuth [Knu97, Section 4.6.3, Exercise 36] et von zur Gathen et Gerhard [GG03, Exercise 14.30].

La preuve du Théorème 18.10 utilise l'existence de la factorisation irréductible. Pour une autre preuve constructive le lecteur pourra consulter celle du livre de Mines *et alii* [MRR88, Chapter VI, Theorem 6.3].

Le fait qu'il existe des corps effectifs de caractéristique non nulle tels que l'extraction de racines p-ièmes ne soit pas calculable a été établi par Fröhlich et Shepherdson en 1956 [FS56, Section 7]. Une autre preuve est esquissée par von zur Gathen [Gat84, Remark 5.10].

Bibliographie

FS56 Fröhlich, A. et J. C. Shepherdson (1956). « Effective procedures in field theory ». In : *Philosophical Transactions of the Royal Society A*, vol. 248, p. 407–432.

Gat06 Gathen, Joachim von zur (2006). « Who was who in polynomial factorization ». In : *ISSAC'06 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 1–2.

Gat84 — (1984). « Hensel and Newton methods in valuation rings ». In : *Mathematics of Computation*, vol. 42, n°166, p. 637–661.

GG03 Gathen, Joachim von zur et Jürgen Gerhard (2003). *Modern computer algebra*. 2e éd. Cambridge University Press.

GT96 Gianni, P. et B. Trager (1996). « Square-free algorithms in positive characteristic ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 7, n°1, p. 1–14.

Knu97 Knuth, Donald E. (1997). *The art of computer programming*. 3e éd. Vol. 2 : Seminumerical Algorithms. Computer Science and Information Processing. Addison-Wesley Publishing Co.

Lec08 Lecerf, Grégoire (2008). « Fast separable factorization and applications ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 19, n°2, p. 135–160.

Lec13 — (2013). « Factorisation des polynômes à plusieurs variables (Cours no. II) ». In : *Journées nationales de calcul formel*. Éd. par G. Chèze, P. Boito, C. Pernet et M. Safey el Din. Vol. 3. Les cours du CIRM n°1. Cedram, p. 1–85. url : http://ccirm.cedram.org/ccirm-bin/fitem?id=CCIRM_2013__3_1_A2_0.

MRR88 Mines, Ray, Fred Richman et Wim Ruitenburg (1988). *A course in constructive algebra*. Universitext. Springer-Verlag.

Mus71 Musser, David R. (1971). « Algorithms for polynomial factorization ». Thèse de doctorat. Univ. of Wisconsin : C.S. Department.

Ste05 Steel, Allan (2005). « Conquering inseparability : primary decomposition and multivariate factorization over algebraic function fields of positive characteristic ». In : *Journal of Symbolic Computation*, vol. 40, n°3, p. 1053–1075.

Yun76 Yun, David Y. Y. (1976). « On square-free decomposition algorithms ». In : *SYMSAC'76 : ACM Symposium on Symbolic and Algebraic Computation*. Yorktown Heights, New York, United States : ACM, p. 26–35.

19. Factorisation des polynômes à une variable sur un corps fini

Résumé

Les algorithmes de factorisation des polynômes dépendent fortement du corps de leurs coefficients. Il est alors naturel de commencer avec le cas des corps finis. Dans ce chapitre nous présentons les algorithmes historiques qui sont encore très largement utilisés en pratique. Grâce aux résultats du chapitre précédent nous pouvons supposer que le polynôme à factoriser est séparable.

19.1 Corps finis, quelques rappels

Nous commençons par rappeler quelques résultats élémentaires de théorie des groupes dont nous avons besoin.

Préliminaires : groupes commutatifs finis, arithmétique

Soit G un groupe commutatif fini, noté multiplicativement (dans la suite, G sera le groupe $\mathrm{F}_q^* = \mathbb{F}_q \setminus \{0\}$, $\mathbb{F}_q$ étant un corps fini à q éléments). L'*ordre* d'un élément a de G est le plus petit entier positif non nul γ tel que $a^\gamma = 1$. Dans la suite nous utiliserons les résultats suivants :

Lemme 19.1 L'ordre de tout élément divise le cardinal card(G) de G, et donc $a^{\mathrm{card(G)}} = 1$ pour tout a de G.

Démonstration. L'ensemble des puissances de a est un sous-groupe de G, de cardinal γ. On utilise alors le théorème de Lagrange, qui dit que le cardinal d'un sous-groupe de G divise card(G). ■

Lemme 19.2 Soit γ le ppcm des ordres des éléments de G. Alors, il existe un élément d'ordre γ ; en particulier, γ divise card(G).

Lemme 19.3 Soient p, m, n trois entiers positifs, avec $p \geq 2$. Si $p^m - 1$ divise $p^n - 1$ alors m divise n.

Démonstration. En écrivant $n = qm + r$, avec $0 \leq r < m$, nous avons $p^r = p^n \bmod (p^m - 1)$, donc $p^r = 1 \bmod (p^m - 1)$. Puisque $r < m$, nous déduisons que $p^r = 1$ et donc $r = 0$. ■

Corps finis

L'exemple le plus simple de corps fini, noté $\mathbb{F}_p$, est un quotient de la forme $\mathbb{Z}/p\mathbb{Z}$, où p est un nombre premier. Attendu qu'un tel corps ne contient pas de sous-corps propre, il est dit *premier*.

Soit ensuite P un polynôme irréductible de $\mathbb{F}_p[X]$, de degré d. Alors, $\mathbb{K} = \mathbb{F}_p[X]/(P)$ est également un corps ; puisque $1, X, \ldots, X^{d-1}$ forment une base de $\mathbb{K}$ comme espace vectoriel sur $\mathbb{F}_p$, le cardinal de $\mathbb{K}$ est p^d. Si nous prenons un autre polynôme irréductible P de degré d nous obtenons un autre corps fini isomorphe à $\mathbb{K}$. Un corps fini à $q = p^d$ éléments est noté $\mathbb{F}_q$. En admettant l'existence de polynômes irréductibles de n'importe quel degré à coefficients dans $\mathbb{F}_p$, il est aisé de construire n'importe quel corps fini de cardinal p^d quelle que soit la valeur de d. La proposition suivante montre que $\mathbb{F}_q$ correspond à l'ensemble des racines de $X^q - X$. Le *groupe multiplicatif* des éléments non nuls de $\mathbb{F}_q$ est noté $\mathbb{F}_q^*$. Son cardinal est $q - 1$.

Proposition 19.4 Pout tout a dans $\mathbb{F}_q^*$, nous avons $a^{q-1} = 1$ et $X^q - X = \prod_{a \in \mathbb{F}_q} (X - a)$.

Démonstration. Pour tout a dans $\mathbb{F}_q^*$, le Lemme 19.1 nous donne $a^{q-1} = 1$. Par conséquent $a^q = a$ pout tout a de $\mathbb{F}_q$, nul ou pas. Ceci montre que le polynôme $X^q - X$ s'annule sur tous les éléments de $\mathbb{F}_q$, donc $\prod_{a \in \mathbb{F}_q} (X - a)$ divise $X^q - X$. Comme les deux polynômes sont unitaires et de même degré, ils sont égaux. ■

Corollaire 19.5 Pour tout $i = 1, \ldots, p - 1$, le coefficient binomial $\binom{p}{i}$ est divisible par p.

Démonstration. Rappelons que ce coefficient binomial est le coefficient de X^i dans $(X + 1)^p$. La proposition précédente entraîne les égalités

$$\begin{aligned} X^p - X &= \prod_{a \in \mathbb{F}_p} (X - a) = \prod_{a \in \mathbb{F}_p} (X - a + 1) \\ &= \prod_{a \in \mathbb{F}_p} ((X + 1) - a) = (X + 1)^p - (X + 1). \end{aligned}$$

On en déduit que $(X + 1)^p = X^p + 1$, qui est ce que l'on voulait montrer. ■

Le corollaire suivant est, d'une manière ou d'une autre, à la base de la plupart des résultats « arithmétiques » sur les corps finis.

Corollaire 19.6 Soit F un polynôme quelconque dans $\mathbb{F}_q[X]$ et soit A le quotient $\mathbb{F}_q[X]/(F)$ (qui n'est pas nécessairement un corps). L'application, dite de *Frobenius*,

$$\phi_F : \begin{array}{ccc} A & \to & A \\ a & \mapsto & a^q \end{array}$$

est $\mathbb{F}_q$-linéaire.

Démonstration. Soient λ dans $\mathbb{F}_q$ et a, b dans A. L'égalité $\phi_F(\lambda a) = \lambda\phi(a)$ est une conséquence de la Proposition 19.4, appliquée à $\mathbb{F}_q$. L'égalité $\phi_F(a+b) = \phi_F(a) + \phi_F(b)$ est une conséquence du corollaire précédent, en développant le produit $(a+b)^q$. ■

Voyons quelques conséquences de ce corollaire.

Proposition 19.7 Pour tous $c_0, \ldots, c_n$ dans $\mathbb{F}_q$, nous avons l'égalité suivante :

$$\left(\sum_{i=0}^{n} c_i X^i\right)^q = \sum_{i=0}^{n} c_i X^{iq}.$$

Démonstration. Il s'agit d'une conséquence facile de la Proposition 19.4 et du Corollaire 19.5. ■

Proposition 19.8 Pour tout $e \geq 1$, le polynôme $X^{q^e} - X$ est le produit des polynômes unitaires et irréductibles de $\mathbb{F}_q[X]$ dont le degré divise e.

Démonstration. Soit F un polynôme irréductible de $\mathbb{F}_q[X]$ de degré n divisant e, et soit $\mathbb{K}$ le quotient $\mathbb{F}_q[X]/(F)$. Soit ensuite x la classe de X dans $\mathbb{K}$; la Proposition 19.4 appliquée à x montre que $x^{q^n-1} = 1$ donc $x^{q^n} = x$, d'où $x^{q^e} = x$. Par conséquent F divise $X^{q^e} - X$.

Réciproquement, soit F un facteur irréductible de degré n de $X^{q^e} - X$; il faut montrer que n divise e. Soit $\mathbb{K}$ le quotient $\mathbb{F}_q[X]/(F)$ et soit x l'image de X dans $\mathbb{K}$. Puisque $F(x) = 0$, alors $x^{q^e} - x$ est nul. Pour tous λ_i dans $\mathbb{F}_q$, nous avons alors les égalités suivantes :

$$\left(\sum_i \lambda_i x^i\right)^{q^e} = \sum_i \lambda_i (x^i)^{q^e} = \sum_i \lambda_i (x^{q^e})^i = \sum_i \lambda_i x^i,$$

la première égalité s'obtenant en appliquant n fois le Corollaire 19.6. Donc tout élément de $\mathbb{K}$ est racine de $X^{q^e} - X$, de sorte que tout élément non nul de $\mathbb{K}$ est racine de $X^{q^e-1} - 1$.

Soit maintenant γ l'exposant du groupe $\mathbb{K} \setminus \{0\}$, c'est-à-dire le ppcm des ordres de ses éléments ; d'après le Lemme 19.2, γ divise $\mathrm{card}(\mathbb{K} \setminus \{0\}) = q^n - 1$. Par ailleurs, par définition, tout élément non nul de $\mathbb{K}$ annule $X^\gamma - 1$, ce qui implique $\gamma \geq q^n - 1$, et donc $\gamma = q^n - 1$. Le Lemme 19.2 montre ensuite qu'il existe un élément a d'ordre $q^n - 1$

dans $\mathbb{K} \setminus \{0\}$. De $a^{q^e-1} = 1$, on déduit que $q^n - 1$ divise $q^e - 1$. Le Lemme 19.3 permet alors de conclure que n divise e.

À ce stade, on sait donc que les facteurs irréductibles de $X^{q^e} - X$ sont exactement les polynômes irréductibles de degré divisant e. Il reste à montrer qu'il n'existe pas de facteur multiple. Pour cela, il suffit de constater que la dérivée de $X^{q^e} - X$ vaut -1, et en particulier que son pgcd avec $X^{q^e} - X$ vaut 1. Il suffit pour finir de faire appel à la Proposition 18.2. ■

19.2 Algorithme de Berlekamp

Nous commençons par présenter des algorithmes de factorisation, attribués à Berlekamp. À partir de maintenant, le polynôme F à factoriser est supposé unitaire, séparable, de degré n, et a ses coefficients dans le corps fini $\mathbb{F}_q$, avec $q = p^d$. Nous noterons $F_1, \ldots, F_r$ les facteurs irréductibles unitaires de F, que nous cherchons à calculer, de sorte que $F = F_1 \cdots F_r$.

Réduction à l'algèbre linéaire

En vertu du Corollaire 19.6, l'application

$$\begin{array}{rccc} \psi_F : & \mathbb{F}_q[X]/(F) & \to & \mathbb{F}_q[X]/(F) \\ & G & \mapsto & G^q - G \end{array}$$

est une application $\mathbb{F}_q$-linéaire. L'étude du noyau de ψ_F va permettre de factoriser F, à partir de la proposition suivante :

Proposition 19.9 Un polynôme G de degré au plus $n-1$ est dans le noyau de ψ_F si, et seulement si, G est une constante modulo chaque F_i.

Démonstration. Par la Proposition 19.4, le polynôme $X^q - X$ se factorise dans $\mathbb{F}_q$, et donc dans $\mathbb{F}_q[X]/(F_i)$, sous la forme $\prod_{a \in \mathbb{F}_q}(X - a)$. Puisque $\mathbb{F}_q[X]/(F_i)$ est un corps contenant $\mathbb{F}_q$, un élément $b \in \mathbb{F}_q[X]/(F_i)$ est une racine de $X^q - X$ si et seulement s'il annule un des facteurs $X - a$, donc si, et seulement si, $a \in \mathbb{F}_q$. ■

En notant $\hat{F}_i = F/F_i$, nous pouvons maintenant observer que les

$$B_i = (\hat{F}_i F_i'/F') \bmod F$$

pour i de 1 à r satisfont la propriété suivante : $B_i \bmod F_j$ vaut 1 si $j = i$ et 0 sinon. Il s'ensuit que les B_i forment une base du noyau de ψ_F. Un polynôme G dans le noyau de ψ_F est donc nécessairement une combinaison linéaire sur $\mathbb{F}_q$ de la forme $G = \sum_{i=1}^{n} b_i B_i$. La proposition précédente donne déjà un moyen de calculer le nombre r de facteurs irréductibles de F. La proposition suivante donne une méthode pour en déduire un scindage non trivial de F.

Proposition 19.10 Si G est non constant dans le noyau de ψ_F, alors les constantes $G \bmod F_i$ se sont pas toutes identiques.

Entrée F séparable de degré n dans $\mathbb{F}_q[X]$.
Sortie Un facteur non trivial de F si F n'est pas irréductible.

1. Calculer la matrice de l'application ψ_F.
2. Si le rang de ψ_F est $n-1$ alors renvoyer le fait que F est irréductible.
3. Choisir un polynôme G de degré au plus $n-1$ et non constant dans le noyau de ψ_F.
4. Calculer les $\gcd(F, G-a)$ pour toutes les valeurs de a dans $\mathbb{F}_q$, jusqu'à trouver un scindage non trivial de F.

Algorithme 19.1 – Algorithme déterministe de Berlekamp.

Démonstration. Puisque tous les F_i sont distincts et irréductibles, par le théorème des restes chinois, nous avons un isomorphisme

$$\mathbb{F}_q[X]/(F) = \mathbb{F}_q[X]/(F_1) \times \cdots \times \mathbb{F}_q[X]/(F_r),$$

donné par

$$G \bmod F \mapsto (G \bmod F_1, \ldots, G \bmod F_r).$$

Les F_i étant irréductibles, chaque $\mathbb{F}_q[X]/(F_i)$ est un corps. Dans la représentation « produit de corps », l'application ψ_F s'écrit bien sûr :

$$(G_1, \ldots, G_r) \mapsto (G_1^q - G_1 \bmod F_1, \ldots, G_r^q - G_r \bmod F_r).$$

Si tous les $G_i = G \bmod F_i$ sont les mêmes, alors nécessairement $G = G_i$ pout tout i, car le degré de G est plus petit que F. En particulier, G devrait être constant, ce qui est contraire à nos hypothèses. ■

Algorithme déterministe

La clé de l'algorithme déterministe de Berlekamp réside dans ce résultat :

Proposition 19.11 Soit G comme dans la proposition précédente. Il existe a dans $\mathbb{F}_q$ tel que $\gcd(F, G-a)$ est un diviseur non trivial de F.

Démonstration. Pour a quelconque dans $\mathbb{F}_q$, nous avons l'égalité $\gcd(F, G-a) = \prod_i F_i$, où le produit est pris sur tous les F_i tels que F_i divise $G-a$, c'est-à-dire tels que $G \bmod F_i = a$. Puisqu'il existe au moins deux valeurs distinctes pour les $G \bmod F_i$ et que chacune d'entre elles donne un diviseur de F, ces diviseurs sont stricts. ■

Dans la suite de ce chapitre nous notons $\theta > 2$ un exposant réalisable pour la multiplication de matrices sur $\mathbb{F}_q$, tel que défini dans le Chapitre 8.

Proposition 19.12 L'Algorithme 19.1 calcule un facteur non trivial de F au moyen d'au plus $O(n^\theta + (q+n)M(n)\log n)$ opérations arithmétiques dans $\mathbb{F}_q$.

Démonstration. Le premier pas consiste à calculer la matrice de l'application ψ_F dans la base $1, X, \ldots, X^{n-1}$. On commence par calculer $X^q \bmod F$: cela demande $O(\log q)$ opérations modulo F, soit $O(M(n)\log q)$ opérations dans $\mathbb{F}_q$. Ensuite, il faut calculer $(X^2)^p = (X^p)^2, (X^3)^p = (X^p)^3, \ldots$, ce qui se fait en n multiplications modulo F, soit $O(nM(n))$ opérations en tout.

L'algèbre linéaire coûte $O(n^\theta)$. Une fois obtenu un vecteur du noyau, chaque essai de pgcd coûte $O(M(n)\log n)$ opérations, et il faut en faire au plus q. ■

Pour trouver la factorisation complète, il suffit d'itérer le procédé, ce qui entraîne *a priori* un surcoût de n. En fait, il n'y a pas besoin de refaire de l'algèbre linéaire, mais seulement de parcourir tous les vecteurs d'une base du noyau de ψ_F, de sorte que le coût total de la factorisation est

$$O(n^\theta + qnM(n)\log n)$$

opérations de $\mathbb{F}_q$. Nous avons alors montré le résultat suivant :

Proposition 19.13 Un polynôme F séparable de $\mathbb{F}_q[X]$ de degré n peut être factorisé en effectuant au plus $O(n^\theta + qnM(n)\log n)$ opérations de $\mathbb{F}_q$.

La complexité de cet algorithme est linéaire en q, ce qui le rend rapidement impraticable, dès lors que q atteint quelques milliers. Par ailleurs, supposer F séparable n'est pas restrictif ici puisque les techniques du chapitre précédent permettent de se ramener facilement à ce cas.

À l'heure actuelle, on ne connaît pas d'algorithme de factorisation déterministe de complexité polynomiale, c'est-à-dire en n et $\log q$. Nous présenterons dans la suite des algorithmes probabilistes de coûts moyens polynomiaux.

Algorithme probabiliste

L'algorithme probabiliste de Berlekamp repose essentiellement sur les même ingrédients que la version déterministe précédente. Supposons p impair, c'est-à-dire autre que 2. L'entier $q-1$ est alors divisible par 2 et la puissance $(q-1)/2$ d'un élément de $\mathbb{F}_q$ vaut soit -1 soit 1. Sur l'ensemble des éléments non nuls il y a autant de telles puissances valant -1 que 1. Donc si G est pris comme une combinaison linéaire non nulle des vecteurs d'une base fixe du noyau de ψ_F, alors on peut attendre que

$$G^{(q-1)/2} = \sum_{i=1}^{r} b_i^{(q-1)/2} B_i^{(q-1)/2}$$

ait quelques possibles valeurs $b_i^{(q-1)/2}$ qui valent 0, et la plupart des autres qui valent -1 ou 1 à peu près en même proportion. Du coup, il est possible que $\mathrm{pgcd}(G, F)$ soit un facteur non trivial de F, mais en général ce n'est pas le cas et on examine $\mathrm{pgcd}(G^{(q-1)/2} - 1, F)$. Ce dernier polynôme est souvent un facteur non trivial de F contenant une quantité proche de $r/2$ facteurs irréductibles en moyenne. Nous formaliserons davantage cette description de l'algorithme dans les paragraphes suivants. Lorsque $p = 2$, l'entier $q-1$ n'est pas divisible par 2 et la technique que nous venons

Entrée $F \in \mathbb{F}_q[x]$ séparable de degré n, et une base $G_1, \ldots, G_r$ du noyau de l'endomorphisme de Frobenius ψ_F.
Sortie Les facteurs irréductibles $F_1, \ldots, F_r$ de F.

1. Si $r = 1$ alors renvoyer F.
2. Choisir des éléments $g_1, \ldots, g_r$ dans $\mathbb{F}_q$ à l'aide d'un générateur aléatoire, et calculer $G = g_1 G_1 + \cdots + g_r G_r$.
3. Si $H_1 = \mathrm{pgcd}(F, G)$ n'est pas constant alors aller à l'étape (6).
4. Calculer $H = G^{\frac{q-1}{2}} - 1 \bmod F$ si p est impair, et $H = \mathrm{Tr}_d(G) \bmod F$ si $p = 2$.
5. Si $H_1 = \mathrm{pgcd}(F, H)$ est constant alors retourner à l'étape (2).
6. Calculer $H_2 = F/H_1$.
7. Calculer une base $G_{1,1}, \ldots, G_{r_1,1}$ des $(G_i F' \bmod F)/(H_1' H_2) \bmod H_1$ pour $i \in \{1, \ldots, r\}$.
8. Calculer une base $G_{1,2}, \ldots, G_{r_2,2}$ des $((G_i F' \bmod F)/(H_1 H_2') \bmod H_2$ pour $i \in \{1, \ldots, r\}$.
9. Renvoyer la réunion des facteurs obtenus par les appels récursifs avec $H_1, G_{1,1}, \ldots, G_{r_1,1}$, d'une part, puis $H_2, G_{1,2}, \ldots, G_{r_2,2}$ d'autre part.

Algorithme 19.2 – Deuxième partie de l'algorithme probabiliste de Berlekamp.

d'ébaucher ne s'applique plus. Il est alors d'usage de faire appel à la construction suivante un peu plus coûteuse en calculs :

Définition 19.1 Si $p = 2$, le polynôme de la d-ième trace sur $\mathbb{F}_2$ est $\mathrm{Tr}_d(X) = X^{2^{d-1}} + X^{2^{d-2}} + \cdots + X^4 + X^2 + X + 1 \in \mathbb{F}_{2^d}[X]$.

Lemme 19.14 Tout $a \in \mathbb{F}_{2^d}$ vérifie $\mathrm{Tr}_d(a) \in \mathbb{F}_2$ et $\mathrm{card}(\mathrm{Tr}_d^{-1}(0)) = \mathrm{card}(\mathrm{Tr}_d^{-1}(1)) = 2^{d-1}$.

Démonstration. Puisque $X^{2^d} + X = \mathrm{Tr}_d^2(X) + \mathrm{Tr}_d(X) = \mathrm{Tr}_d(X)(\mathrm{Tr}_d(X) + 1)$ nous avons bien $\mathrm{Tr}_d(a) \in \mathbb{F}_2$. Nous obtenons ainsi le fait que Tr_d a toutes ses 2^{d-1} racines dans $\mathbb{F}_{2^d}$. ■

Exercice 19.1 Si $p = 2$, si F est un polynôme de degré n dans $\mathbb{F}_q[X]$, et si G est un polynôme de degré au plus $n-1$, montrer que $\mathrm{Tr}_d(G) \bmod F$ peut se calculer au moyen de $d\mathsf{M}(n)$ opérations dans $\mathbb{K}$. ■

La première partie de l'algorithme probabiliste de Berlekamp est le calcul d'une base $G_1, \ldots, G_r$ du noyau de ψ_F. La deuxième partie, résumée dans l'Algorithme 19.2, consiste à calculer tous les facteurs irréductibles de F à partir de cette base. Rappelons que le coût moyen d'un algorithme pour une entrée donnée est la moyenne des coûts sur toutes les exécutions possibles.

Proposition 19.15 L'Algorithme 19.2 est correct et nécessite au plus

$$O(n^\theta \log r + (r + \log n + \log q)M(n)\log r)$$

opérations dans $\mathbb{F}_q$ en moyenne.

Démonstration. Si pgcd(F, G) à l'étape (2) n'est pas constant alors il s'agit d'un facteur non trivial de F. Si p est impair, les facteurs irréductibles de H_1 à l'étape (5) sont les F_i tels que $G^{(q-1)/2} \bmod F_i$ vaut 1.

Nous avons vu que chaque r-uplet $(g_1, \ldots, g_r)$ est en bijection avec un r-uplet $(b_1, \ldots, b_r)$ de $\mathbb{F}_q^r$ satisfaisant $G = \sum_{i=1}^r b_i B_i$. Sur les q^r valeurs possibles du r-uplet $(b_1, \ldots, b_r)$, $(q-1)^r$ d'entre elles sont dans $(\mathbb{F}_q^*)^r$. Donc pour $q^r - 1 - (q-1)^r$ un scindage non trivial de F est trouvé dès l'étape (3). Si p est impair, les r-uplets $(b_1^{(q-1)/2}, \ldots, b_r^{(q-1)/2})$ sont équi-distribués dans $\{-1, 1\}^r$. Seuls les deux tels uplets constants sur les 2^r possibles ne conduisent pas à un scindage non trivial de F à l'étape (5). La proportion d'uplets qui ne conduisent pas à un scindage non trivial est donc

$$\frac{1 + (q-1)^r / 2^{r-1}}{q^r} \leq 1/2.$$

Le nombre moyen d'uplets à essayer est donc borné par une constante. Autrement dit, l'algorithme arrive rapidement à l'étape (6) en moyenne.

À partir de l'identité

$$(B_i F' \bmod F)/(H_1' H_2) \equiv (F/F_i)/(H_1' H_2) \equiv (H_1/F_i)H_1' \bmod H_1,$$

nous voyons que $(B_i F' \bmod F)/(H_1' H_2)$ est dans le noyau de ψ_{H_1} pour tout $i \in \{1, \ldots, r\}$, et que les $G_{i,1}$ forment bien un ensemble générateur du noyau de ψ_{H_1}. Il en va de même concernant H_2. Par conséquent l'algorithme fonctionne correctement. Nous laissons le soin au lecteur de traiter le cas particulier où $p = 2$.

L'étape (2) coûte $O(rn)$ opérations. L'étape (4) coûte $O(M(n)\log q)$ opérations. Les étapes (3), (5), et (6) utilisent $O(M(n)\log n)$ opérations. Ensuite, les étapes (7) et (8) nécessitent $O((r + \log n)M(n) + n^\theta)$ opérations en faisant appel au Théorème 8.4. Le nombre d'opérations total en moyenne avant d'entrer dans les appels récursifs est donc $O(n^\theta + (r + \log n + \log q)M(n))$.

Considérons maintenant l'arbre des appels récursifs. Au début, à la racine, le degré du polynôme en entrée est n. Si $r \geq 2$, à la profondeur 1 de cet arbre, il y a deux nœuds correspondant à H_1 et H_2 trouvés par l'algorithme. La somme des degrés de H_1 et H_2 est n. Il est aisé de constater qu'à une profondeur donnée la somme des degrés de tous les polynômes à factoriser est toujours au plus n. De part la super-additivité de M, le coût total des calculs propres à tous les nœuds d'une même profondeur est donc $O(n^\theta + (r + \log n + \log q)M(n))$ opérations dans $\mathbb{F}_q$. Il reste à montrer que la profondeur moyenne de l'arbre des appels récursifs est $O(\log r)$ pour conclure l'analyse de complexité.

Considérons le cas où p est impair. Notons ℓ la variable aléatoire représentant la profondeur de l'arbre, et $E(\ell)$ la moyenne des profondeurs. Si $\mathcal{P}(\ell \geq h)$ représente la

Entrée F séparable dans $\mathbb{F}_q[X]$.
Sortie La factorisation irréductible de F.
1. Calculer, dans la base $1, X, \ldots, X^{n-1}$, la matrice de l'endomorphisme de Frobenius ψ_F.
2. Calculer une base $G_1, \ldots, G_r$ du noyau de ψ_F.
3. Appeler l'Algorithme 19.2 avec $F, G_1, \ldots, G_r$ et renvoyer la factorisation obtenue.

Algorithme 19.3 – Algorithme probabiliste de Berlekamp.

probabilité que la profondeur soit au moins h, alors

$$E(\ell) = \sum_h h(\mathcal{P}(\ell \le h) - \mathcal{P}(\ell \le h+1)) = \sum_h \mathcal{P}(\ell \ge h).$$

La probabilité que deux facteurs $i < j$ qui sont dans le même nœud à la profondeur $h-1$ le reste à la profondeur h est $1/q^2 + 1/2$. Elle correspond aux possibilités suivantes pour $(b_i, b_j) : (0,0)$, $(b_i, b_j) \in (\mathbb{F}_q^*)^2$ avec $b_i^{(q-1)/2} = b_j^{(q-1)/2}$. Par conséquent, la probabilité que deux facteurs $i < j$ soient dans le même nœud jusqu'à la profondeur h est $(1/q^2 + 1/2)^h$. En prenant la somme de toutes ces probabilités sur toutes les paires $i < j$, on déduit que la probabilité $\mathcal{P}(\ell \ge h)$ que tous les facteurs de F n'aient pas été découverts à la profondeur $h-1$ est au plus $r^2(1/q^2 + 1/2)^{h-1} \le r^2(3/4)^{h-1}$. Finalement nous déduisons la borne souhaitée :

$$\sum_h \mathcal{P}(\ell \le h) \le \sum_{h \le (2\log h)/\log(3/4)} \mathcal{P}(\ell \le h) + \sum_{h > (2\log h)/\log(3/4)} \mathcal{P}(\ell \le h) = O(\log r).$$

À nouveau, nous laissons l'étude du cas $p = 2$ en exercice. ■

Nous obtenons finalement une version probabiliste de l'algorithme de Berlekamp dans l'Algorithme 19.3.

Proposition 19.16 L'Algorithme 19.3 est correct et utilise en moyenne

$$O(n^\theta \log n + (n + \log q)M(n)\log n)$$

opérations dans $\mathbb{F}_q$.

Démonstration. Le fait que l'algorithme est correct provient de la Proposition 19.15. Le coût de la première étape est dans $O((n + \log q)M(n))$. L'étape (2) effectue $O(n^\theta)$ opérations par le Théorème 8.4. Le reste du coût provient essentiellement de la proposition précédente. ■

19.3 Algorithme de Cantor et Zassenhaus

Une composante commune à de nombreux algorithmes de factorisation sur les corps finis est la suite des itérés de Frobenius $(\Phi_i)_{i \ge 0}$, relative à F définie par $\Phi_i =$

Entrée F dans $\mathbb{F}_q[X]$ de degré n.
Sortie La factorisation $E_1, \ldots, E_n$ par degrés des facteurs irréductibles de F.
1. Calculer $\Phi_0, \ldots, \Phi_n$.
2. Pour chaque i de 1 à n, calculer $E_i = \text{pgcd}(\Phi_i - X, F)$ puis remplacer F par F/E_i.
3. Renvoyer $E_1, \ldots, E_n$.

Algorithme 19.4 – Algorithme de factorisation par degrés.

$X^{q^i} \bmod F$. D'une façon naïve, les $n+1$ premiers itérés $\Phi_0, \ldots, \Phi_n$ peuvent être calculés en utilisant $O(nM(n)\log q)$ opérations dans $\mathbb{F}_q$.

La dépendance du coût du calcul des itérés de Frobenius peut être diminuée en considérant l'opération de composition modulaire définie comme le calcul de $G \circ H \bmod F$ pour G et H deux polynômes de degré au plus $n-1$. Nous pouvons calculer Φ_1 en $O(M(n)\log q)$ opérations puis obtenir Φ_{i+1} comme $\Phi_1 \circ \Phi_i \bmod F$. Avec cette idée, pour calculer calculer les $n+1$ premiers itérés, il existe une technique spécifique exploitant l'évaluation et interpolation rapide d'un polynôme en plusieurs points du Chapitre 5, et donnant le résultat suivant :

Proposition 19.17 Avec les notations précédentes, une fois Φ_1 connu, les polynômes $\Phi_0, \ldots, \Phi_l$, pour $l \leq n$, peuvent être calculés avec $O(M(n)^2 \log n \log l)$ opérations dans $\mathbb{F}_q$.

Démonstration. L'évaluation de Φ_i en $\Phi_1, \ldots, \Phi_i \bmod F$ fournit les polynômes $\Phi_{i+1}, \ldots, \Phi_{2i}$. Le coût total découle alors de la Proposition 5.6. ■

L'algorithme de Cantor et Zassenhaus comporte deux étapes. La première consiste à décomposer F en un produit de polynômes $\prod_{i \geq 1} E_i$ où chaque E_i ne comporte que des facteurs irréductibles de degré exactement i. Nous parlons alors de *factorisation par degrés* des facteurs. La deuxième étape consiste à factoriser individuellement chaque E_i en irréductibles.

Proposition 19.18 L'Algorithme 19.4 est correct et nécessite

$$O(M(n)(M(n)\log^2 n + \log q))$$

opérations dans $\mathbb{F}_q$.

Démonstration. La formule utilisée pour calculer E_i repose sur la Proposition 19.8. Le coût total provient directement de la proposition précédente. ■

Notons que l'algorithme précédent ne fait pas intervenir de valeurs aléatoires. L'étape suivante consiste à supposer que F est le produit de r polynômes irréductibles de degré exactement n/r.

Entrée F séparable de degré n ayant tous ses facteurs irréductibles de degré n/r.
Sortie La factorisation irréductible $F_1, \ldots, F_r$ de F.

1. Si $r = 1$ alors renvoyer F.
2. Répéter :
   a. choisir un polynôme G de degré au plus $n-1$ d'une façon aléatoire ;
   b. calculer $H_1 = \mathrm{pgcd}(F, G)$. Si H_1 n'est pas constant alors aller à l'étape (3) ;
   c. calculer $H = G^{\frac{q^{n/r}-1}{2}} \bmod F$ si p est impair et $H = \mathrm{Tr}_{kn/r}(G) \bmod F$ si $p = 2$;
   d. calculer $H_1 = \mathrm{pgcd}(F, H-1)$;
   e. si H_1 n'est pas constant alors aller à l'étape (3).
3. Calculer $H_2 = F/H_1$ et appeler récursivement la fonction sur H_1 et H_2 et renvoyer la réunion des facteurs.

Algorithme 19.5 – Algorithme de factorisation en équi-degré.

Proposition 19.19 L'Algorithme 19.5 est correct et effectue

$$O(n\mathsf{M}(n)\log q \log n)$$

opérations dans $\mathbb{F}_q$ en moyenne. Ce coût tombe dans

$$O(\mathsf{M}(n)^2 \log^3 n + \mathsf{M}(n)\log q \log n)$$

lorsque p est impair.

Démonstration. En notant $\hat{F}_i = F/F_i$, nous avons précédemment observé que les $B_i = (\hat{F}_i F_i'/F') \bmod F$ pour i de 1 à r satisfont la propriété suivante : $B_i \bmod F_j$ vaut 1 si $j = i$ et 0 sinon.

Si G est un polynôme de degré au plus $n-1$, il s'écrit d'une unique façon sous la forme $\sum_{i=1}^{r} C_i(X)B_i(X)$ avec $C_i(X) \in \mathbb{F}_q[X]$ de degré au plus $n/r-1$, par le théorème des restes chinois. Du coup $G^{(q^{n/r}-1)/2} \equiv C_i^{(q^{n/r}-1)/2} \bmod F_i$ vaut assez rarement 0, et plus fréquemment soit -1 soit 1 dans la même proportion. Donc H_1 définit bien un scindage non trivial de F en général. Le résultat renvoyé par l'algorithme est donc correct. Nous laissons le soin au lecteur d'adapter la preuve de l'algorithme probabiliste de Berlekamp pour prouver que la profondeur moyenne des appels récursifs est dans $O(\log n)$.

Le coût du calcul de H à l'étape (2) peut se faire en $O(n\mathsf{M}(n)\log q)$. Les autres opérations totalisent un coût dans $O(\mathsf{M}(n)\log n)$. Lorsque p est impair il est intéressant de procéder comme suit. En notant $m = n/r$ et $\alpha = H \bmod F$, comme

$$\frac{q^m - 1}{2} = (q^{m-1} + q^{m-2} + \cdots + q + 1)\frac{q-1}{2},$$

nous pouvons calculer $\alpha^{\frac{q^m-1}{2}} = (\alpha^{q^{m-1}} \cdots \alpha^q \alpha)^{\frac{m-1}{2}}$ grâce à la proposition 19.17. ■

Nous arrivons au résultat principal de cette section :

Théorème 19.20 Un polynôme de degré n dans $\mathbb{F}_q[x]$ peut être factorisé avec un nombre moyen d'opérations dans $\mathbb{F}_q$ dans $\tilde{O}(n^2 \log q)$. Ce coût tombe dans $\tilde{O}(n^2 + n \log q)$ si p est impair.

Démonstration. Il s'agit d'une conséquence des Propositions 19.18 et 19.19, en utilisant une fonction de multiplication $M(n)$ dans $\tilde{O}(n)$. ■

Notes

La présentation choisie ici est inspirée du Chapitre 14 du livre de von zur Gathen et Gerhard [GG03]. Le calcul des itérés de Frobenius constitue une brique importante en ce qui concerne les performances pratiques. Les algorithmes décrits dans ce chapitre sont essentiellement ceux utilisés en pratique à ce jour. Ils peuvent être encore un peu améliorés et adaptés pour la recherche des racines, pour tester l'irréductibilité, ou bien encore pour construire des polynômes irréductibles de degré donné. Les analyses détaillées des aspects probabilistes ont d'abord été effectuées par Flajolet et Steyaert [FS82], puis complétées par Flajolet, Gourdon, et Panario [FGP01]. Comme autres livres sur les corps finis et leurs applications, le lecteur pourra être intéressé par consulter celui de Mullen et Panario [MP13], ou bien celui de Shoup [Sho09].

Les premières techniques connues de factorization des polynômes à coefficients dans un corps fini remontent aux travaux de Gauss en 1798, Galois en 1830, et Arwins en 1918. Les premiers algorithmes performants ont été formalisés par Berlekamp [Ber67 ; Ber70], Zassenhaus [Zas69], puis Cantor et Zassenhaus [CZ81]. Ils regroupent les principales idées mathématiques que nous avons présentées ici.

En 1998, Kaltofen et Shoup [KS98] ont relié le problème de factorisation à celui de la composition modulaire, et ont obtenu une borne de complexité en $O(d^{1,815} \log q)$, en terme du nombre d'opérations effectuées dans $\mathbb{F}_q$. Sur un corps fini, Kedlaya et Umans ont ensuite conçu un algorithme de composition modulaire ayant un coût binaire quasi-linéaire [KU08], permettant de déduire un algorithme de factorisation sur $\mathbb{F}_q[X]$ ayant un coût binaire moyen dans $(d^{1,5} + d \log q)^{1+o(1)} \log q$, ce qui représente la meilleure borne de complexité asymptotique connue à ce jour. Malheureusement cette approche ne semble pas plus efficace que l'algorithme de Cantor et Zassenhaus pour les tailles de problèmes que l'on peut traiter avec les ordinateurs actuels.

La conception d'algorithmes déterministes en temps polynomial est toujours un sujet de recherche actif. Un algorithme de coût sous-exponentiel a été conçu par Evdokimov en 1994 [Evd94]. Des solutions partielles sont données dans des cas particuliers, selon les propriétés du polynômes ou bien du corps fini, dans les travaux de Moenck [Moe77], Camion [Cam83], Schoof [Sch85], von zur Gathen [Gat87], Mignotte et Schnorr [MS88], Rónyai [Rón89 ; Rón92], Shoup [Sho90 ; Sho91b], Huang [Hua91], Źrałek [Źra10], Grenet, van der Hoeven et Lecerf [GHL15a ; GHL15b].

Bibliographie

Ber67 Berlekamp, E. R. (1967). « Factoring polynomials over finite fields ». In : *Bell System Technical Journal*, vol. 46, p. 1853–1859.

Ber70 — (1970). « Factoring polynomials over large finite fields ». In : *Mathematics of Computation*, vol. 24, p. 713–735.

Cam83 Camion, Paul (1983). « A deterministic algorithm for factorizing polynomials of $\mathbf{F}_q[X]$ ». In : *Combinatorial mathematics (Marseille-Luminy, 1981)*. Vol. 75. North-Holland Math. Stud. North-Holland, Amsterdam, p. 149–157.

CZ81 Cantor, David G. et Hans Zassenhaus (1981). « A new algorithm for factoring polynomials over finite fields ». In : *Mathematics of Computation*, vol. 36, n°154, p. 587–592.

Evd94 Evdokimov, Sergei (1994). « Factorization of polynomials over finite fields in subexponential time under GRH ». In : *Algorithmic number theory (Ithaca, NY, 1994)*. Vol. 877. Lecture Notes in Computer Science. Springer-Verlag, p. 209–219.

FGP01 Flajolet, P., X. Gourdon et D. Panario (2001). « The complete analysis of a polynomial factorization algorithm over finite fields ». In : *Journal of Algorithms*, vol. 40, n°1, p. 37–81.

FS82 Flajolet, Philippe et Jean-Marc Steyaert (1982). « A branching process arising in dynamic hashing, trie searching and polynomial factorization ». In : *Automata, Languages and Programming*. Éd. par M. Nielsen et E. Schmidt. Vol. 140. Lecture Notes in Computer Science. Springer-Verlag, p. 239–251.

Gat87 Gathen, Joachim von zur (1987). « Factoring polynomials and primitive elements for special primes ». In : *Theoretical Computer Science*, vol. 52, n°1-2, p. 77–89.

GG03 Gathen, Joachim von zur et Jürgen Gerhard (2003). *Modern computer algebra*. 2e éd. Cambridge University Press.

GHL15a Grenet, Bruno, Joris van der Hoeven et Grégoire Lecerf (2015). « Deterministic root finding over finite fields using Graeffe transforms ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 27, n°3, p. 237–257.

GHL15b — (2015). « Randomized root finding over finite FFT-fields using tangent Graeffe transforms ». In : *ISSAC'15 : International Symposium on Symbolic and Algebraic Computation*. Éd. par D. Robertz. ACM Press, p. 197–204.

Hua91 Huang, Ming-Deh A. (1991). « Generalized Riemann hypothesis and factoring polynomials over finite fields ». In : *Journal of Algorithms*, vol. 12, n°3, p. 464–481.

KS98 Kaltofen, Erich et Victor Shoup (1998). « Subquadratic-time factoring of polynomials over finite fields ». In : *Mathematics of Computation*, vol. 67, n°223, p. 1179–1197.

KU08 Kedlaya, Kiran S. et Christopher Umans (2008). « Fast modular composition in any characteristic ». In : *FOCS'08 : IEEE Conference on Foundations of Computer Science*. Washington, DC, USA : IEEE Computer Society, p. 146–155.

Moe77 Moenck, Robert T. (1977). « On the efficiency of algorithms for polynomial factoring ». In : *Mathematics of Computation*, vol. 31, p. 235–250.

MP13 Mullen, Gary L. et Daniel Panario (2013). *Handbook of finite fields*. Discrete Mathematics and Its Applications. Chapman et Hall/CRC.

MS88 Mignotte, Maurice et Claus Schnorr (1988). « Calcul déterministe des racines d'un polynôme dans un corps fini ». In : *Comptes Rendus des Séances de l'Académie des Sciences. Série I. Mathématique*, vol. 306, n°12, p. 467–472.

Rón89 Rónyai, L. (1989). « Factoring polynomials modulo special primes ». In : *Combinatorica. An International Journal on Combinatorics and the Theory of Computing*, vol. 9, n°2, p. 199–206.

Rón92 Rónyai, Lajos (1992). « Galois groups and factoring polynomials over finite fields ». In : *SIAM Journal on Discrete Mathematics*, vol. 5, n°3, p. 345–365.

Sch85 Schoof, René (1985). « Elliptic curves over finite fields and the computation of square roots mod p ». In : *Mathematics of Computation*, vol. 44, n°170, p. 483–494.

Sho09 Shoup, Victor (2009). *A computational introduction to number theory and algebra*. 2e éd. Cambridge University Press.

Sho90 — (1990). « On the deterministic complexity of factoring polynomials over finite fields ». In : *Information Processing Letters*, vol. 33, n°5, p. 261–267.

Sho91b — (1991). « A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic ». In : *ISSAC'91 : International Symposium on Symbolic and Algebraic Computation*. Éd. par S. M. Watt. ACM Press, p. 14–21.

Zas69 Zassenhaus, Hans (1969). « On Hensel factorization I ». In : *Journal of Number Theory*, vol. 1, n°1, p. 291–311.

Źra10 Źrałek, Bartosz (2010). « Using partial smoothness of $p-1$ for factoring polynomials modulo p ». In : *Mathematics of Computation*, vol. 79, p. 2353–2359.

20. Réduction de réseaux et algorithme LLL

Résumé

La recherche de vecteurs courts dans un réseau est la clé de voûte des algorithmes modernes de factorisation des polynômes à coefficients rationnels en temps polynomial. Et plus généralement la réduction des réseaux joue désormais un rôle majeur dans de nombreuses applications. Nous consacrons ce chapitre à l'algorithme classique LLL, nommé d'après les initiales de ses concepteurs A. Lenstra, H. Lenstra et L. Lovász.

20.1 Réseaux et vecteurs courts

On appelle *réseau* de $\mathbb{Z}^n$ un $\mathbb{Z}$-module $\sum_{i=1}^{n} \mathbb{Z}v_i$ engendré par des vecteurs v_i linéairement indépendants sur $\mathbb{Z}$. Il n'est habituellement pas requis que le nombre de tels générateurs corresponde à la dimension n. Néanmoins, ce cas particulier étant suffisant pour les applications de cet ouvrage, nous n'aborderons pas le cas général.

Ce chapitre est consacré au problème du calcul de vecteurs « courts » dans un réseau donné : par exemple, le réseau engendré par les vecteurs $(6,4)$ et $(8,4)$ contient le vecteur plus court $(2,0)$, comme illustré dans la Figure 20.1. Ici, court s'entend pour la norme euclidienne définie par $\|x\| = (x_1^2 + \cdots + x_n^2)^{1/2}$, pour $x \in \mathbb{R}^n$.

La recherche d'un vecteur de norme minimale dans un réseau donné s'avère être un problème difficile, et plus précisément NP-difficile. Il est donc naturel de relâcher la contrainte de minimalité et de considérer le problème approché à un facteur près. Malheureusement, en tolérant un facteur constant tel que $\sqrt{2}$, cette recherche reste NP-difficile. Le problème devient polynomial dès qu'on autorise le facteur d'approximation à croître suffisamment avec la dimension n du réseau.

Ce chapitre est consacré à l'algorithme LLL qui, pour une base $v_1, \ldots, v_n$ d'un réseau L, calcule un vecteur u approximant les plus courts vecteurs du réseau à un

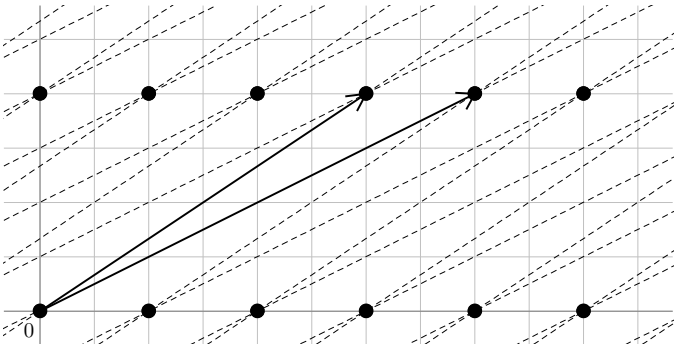


FIGURE 20.1 – Réseau engendré par les vecteurs $(6,4)$ et $(8,4)$.

facteur au plus $2^{(n-1)/2}$.

Théorème 20.1 Donnée une base $v_1,\dots,v_n$ d'un réseau $\mathrm{L} \subseteq \mathbb{Z}^n$, on peut calculer un vecteur u de ce réseau satisfaisant

$$\|u\| \leq 2^{(n-1)/2} \min\{\|v\| \mid v \in \mathrm{L},\ v \neq 0\},$$

en $\tilde{\mathrm{O}}(n^5 \log^2 \mathrm{A})$ opérations binaires, où $\mathrm{A} = \max_{1\leq i\leq n} \|v_i\|$.

L'utilisation de l'algorithme LLL à la factorisation des polynômes à coefficients rationnels fera l'objet du chapitre suivant. Mentionnons ici d'autres applications de nature historique :

— la mise en évidence de la faiblesse de générateurs pseudo-aléatoires ;
— le cassage d'un crypto-système basé sur le problème du sac-à-dos ;
— la recherche de dépendances linéaires entre nombres donnés par des approximations numériques.

Dans les paragraphes suivants nous détaillons ces deux dernières applications d'une façon informelle.

Cassage du crypto-système de Merkle et Hellman

Merkle et Hellman ont proposé en 1978 un crypto-système basé sur le problème du *sac-à-dos*. Ce crypto-système devait nécessiter des calculs moins lourds que le désormais célèbre système RSA.

L'idée de Merkle et Hellman est la suivante. Bob se dote d'une clé secrète constituée d'une famille d'entiers positifs $b_1,\dots,b_n$, telle que chaque b_i soit supérieur à la somme des précédents, d'un multiplicateur c, et d'un autre entier $m > b_1 + \cdots + b_n$. Il rend public les restes positifs a_i de cb_i modulo m, pour chaque $1 \leq i \leq n$. Quand Alice veut lui envoyer le message constitué de la suite de bits $(x_1,\dots,x_n)$, elle construit la somme $s = x_1a_1 + \cdots + x_na_n$ qui constitue le message chiffré. Bob n'a plus qu'à multiplier par l'inverse de c modulo m pour obtenir $x_1b_1 + \cdots + x_nb_n$, et en déduire les x_i pour i décroissant de n à 1.

Le problème sur lequel s'appuie ce crypto-système, celui du sac-à-dos, consiste

précisément à décider l'existence des $x_i \in \{0,1\}$ tels que $s = x_1 a_1 + \cdots + x_n a_n$. Ce problème est NP-complet en l'absence d'hypothèse sur la famille des a_i. Mais dans le cas présent, l'origine des a_i crée une faiblesse cryptographique. Considérons les vecteurs $(0,\ldots,0,1,0,\ldots,0,-a_i) \in \mathbb{Z}^{n+1}$, où 1 est en i-ième position, pour $i \in \{1,\ldots,n\}$, ainsi que le vecteur $(0,\ldots,0,s)$. Tous ces vecteurs sont « longs », puisque de norme de l'ordre de m, mais le réseau qu'ils engendrent contient le vecteur $(x_1,\ldots,x_n,0)$, lequel est manifestement très court. Pour des valeurs pratiques de n, l'algorithme LLL permet donc de déchiffrer les messages assez rapidement.

Relations de dépendance entre constantes numériques

Étant donnés n nombres réels non nuls, $r_1,\ldots,r_n$, une relation de dépendance linéaire à coefficients entiers entre ces réels est une relation de la forme

$$c_1 r_1 + \cdots + c_n r_n = 0$$

pour des coefficients c_i entiers non tous nuls. Lorsqu'on se donne une approximation rationnelle de chaque réel r_i, ou mieux, la possibilité d'obtenir autant de chiffres décimaux que souhaité, la recherche des vecteurs courts dans un réseau permet la détermination de relations de dépendance linéaire entre les r_i. Pour cela, on considère les combinaisons linéaires à coefficients entiers des vecteurs lignes de la matrice

$$\mathrm{V} = \begin{pmatrix} 1 & 0 & \ldots & 0 & a_1 \\ 0 & 1 & \ldots & 0 & a_2 \\ & & \ddots & & \vdots \\ 0 & 0 & \ldots & 1 & a_{n-1} \\ 0 & 0 & \ldots & 0 & a_n \end{pmatrix},$$

où chaque a_i est une troncature entière de $\mathrm{N}r_i$ pour un grand entier N (par exemple, N est une grande puissance de 10). Un vecteur court est alors de la forme

$$u = (c_1,\ldots,c_{n-1},c_1 a_1 + \cdots + c_n a_n) \simeq \big(c_1,\ldots,c_{n-1},\mathrm{N}(c_1 r_1 + \cdots + c_n r_n)\big).$$

En particulier,

$$|c_1 r_1 + \cdots + c_n r_n| \simeq \left|\mathrm{N}^{-1}(c_1 a_1 + \cdots + c_n a_n)\right| \leq \mathrm{N}^{-1}\|u\|$$

se doit d'être petit, ce qui fait de

$$c_1 r_1 + \cdots + c_n r_n = 0$$

un bon candidat pour une relation de dépendance entre les r_i. Bien qu'un tel argument ne constitue pas une preuve, la preuve formelle de relation entre constantes a dans un bon nombre de cas été grandement facilitée une fois que la bonne relation a pu être découverte expérimentalement par cette méthode.

Donnons un exemple. Disons que des arguments théoriques donnent la certitude que le nombre

$$\alpha = \int_0^\infty \frac{\sqrt{x}\ln^5 x}{(1-x)^5}\,\mathrm{d}x$$

peut se représenter comme l'évaluation en π d'une fonction polynomiale à coefficients rationnels. Il s'agit de trouver une dépendance linéaire entre

$$1, \pi, \ldots, \pi^d, \alpha$$

pour un degré de polynôme d à déterminer. Tout système de calcul formel généraliste connait des approximations pour π et permet de calculer aisément une approximation numérique de α. On prend donc par exemple $a_1 = \mathrm{N} = 10^{20}$, $a_2 = 314159265358979323846 \simeq \mathrm{N}\pi$, …, $a_7 = 96138919357530443703022 \simeq \mathrm{N}\pi^6$, $a_{10} = -166994737192290704961 9 \simeq \mathrm{N}\alpha$ pour construire la matrice V. Les normes des vecteurs lignes de cette matrice sont toutes supérieures à $\mathrm{N} = 10^{20}$. Par l'algorithme LLL, on trouve une base du même réseau de vecteurs, dont le premier vecteur ligne,

$$u = (c_1, \ldots, c_{n-1}, c_1 a_1 + \cdots + c_n a_n) = (0, 0, -120, 0, -140, 0, 15, 66),$$

est de norme inférieure à 200, tous les autres vecteurs de base étant de norme supérieure à 380. Il est aisé de déduire $c_n = -24$, en soustrayant $c_1 a_1 + \cdots + c_{n-1} a_{n-1}$ à la dernière coordonnée de u et en divisant par a_n, d'où la relation linéaire

$$\alpha = \int_0^\infty \frac{\sqrt{x}\ln^5 x}{(1-x)^5}\,\mathrm{d}x = \frac{5\pi^2}{24}(3\pi^4 - 28\pi^2 - 24),$$

que l'on peut ensuite prouver être exacte.

Polynôme minimal de nombres algébriques

Donnons nous une approximation numérique d'un nombre réel r qu'on a de bonnes raisons de penser être une racine d'un polynôme P de degré d à coefficients rationnels. La détermination heuristique de P peut se voir comme la recherche d'une relation de dépendance linéaire sur des approximations numériques rationnelles de

$$1, r, \ldots, r^d.$$

Dès lors qu'on dispose d'une borne supérieure sur le degré d de P, on peut donc employer la méthode précédente. Par exemple, soit à déterminer si

$$r \simeq 0{,}266252646290196114530247765575844548176501286103 95\ldots$$

est algébrique. En testant les relations jusqu'à $d = 6$ et pour $\mathrm{N} = 10^{20}$, on trouve le vecteur court

$$u = (1, 0, 0, -54, 0, 0, -8),$$

de norme inférieure à 60, tous les autres vecteurs calculés étant de normes supérieures à 250. On en déduit alors $c_7 = 54$, ce qui conduit au polynôme $\mathrm{P}(\mathrm{X}) = 54\mathrm{X}^6 - 54\mathrm{X}^3 + 1$.

20.2 Le procédé d'orthogonalisation de Gram–Schmidt

L'algorithme LLL exploite le procédé d'orthogonalisation de Gram–Schmidt qui, partant d'une base $v_1, \ldots, v_n$ d'un espace vectoriel sur $\mathbb{Q}$, calcule une base orthogonale

$v_1^*, \ldots, v_n^*$ du même espace vectoriel par la méthode décrite dans les paragraphes suivants.

Le contexte de cette méthode est celui de l'espace vectoriel $\mathbb{Q}^n$ muni du produit scalaire euclidien $(x, y) \mapsto (x \mid y) = x_1 y_1 + \cdots + x_n y_n$ qui a servi à définir la norme euclidienne par $\|x\|^2 = (x \mid x)$. Rappelons que les vecteurs x et y sont dits *orthogonaux* lorsque leur produit scalaire est nul, et que l'*orthogonal* d'un ensemble $S \subset \mathbb{Q}^n$ est l'ensemble noté $S^\perp$ des vecteurs orthogonaux à tous les éléments de S. On introduit les espaces vectoriels emboîtés $V_i = \mathbb{Q}v_1 \oplus \cdots \oplus \mathbb{Q}v_i$. La projection orthogonale sur V_i est le projecteur linéaire sur V_i parallèlement à $V_i^\perp$. On définit alors v_i^* comme la projection orthogonale de v_i sur $V_{i-1}^\perp$ (donc parallèlement à $(V_{i-1}^\perp)^\perp$, qui n'est autre que V_{i-1}).

Les considérations qui précèdent définissent uniquement les v_i^* à partir des v_i comme suit :

$$v_i^* = v_i - \sum_{j=1}^{i-1} \mu_{i,j} v_j^* \qquad \text{où} \qquad \mu_{i,j} = \frac{(v_i \mid v_j^*)}{(v_j^* \mid v_j^*)}.$$

La quantité $\mu_{i,j}$ correspond à l'unique valeur garantissant l'orthogonalité entre v_i^* et v_j^* quand $j < i$. Après avoir posé $\mu_{i,i} = 1$ et $\mu_{i,j} = 0$ quand $j > i$, on obtient la matrice triangulaire inférieure

$$M = (\mu_{i,j}) = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ \mu_{i,j} & & 1 \end{pmatrix},$$

vérifiant la relation

$$M \begin{pmatrix} v_1^* \\ \vdots \\ v_n^* \end{pmatrix} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}.$$

En particulier, les espaces vectoriels sur $\mathbb{Q}$ engendrés par les v_i d'une part et par les v_i^* d'autre part sont les mêmes.

Soit maintenant un vecteur non nul v du réseau engendré par les v_i, qui s'écrit donc $v = \lambda_1 v_1^* + \cdots + \lambda_n v_n^*$ pour des entiers λ_i non tous nuls. En passant au carré de la norme on obtient

$$\|v\|^2 = \sum_{i=1}^{n} \lambda_i^2 \|v_i^*\|^2 \geq \min_{i \leq i \leq n} \|v_i^*\|^2.$$

À ce stade, nous aurions trouvé des vecteurs parmi les plus courts, si ce n'est que les v_i^* sont généralement à coefficients rationnels et donc hors du réseau. Il n'en reste pas moins que l'algorithme LLL qui va suivre s'appuie sur ce procédé d'orthogonalisation de sorte à contrôler des transformations de la base du réseau. À un niveau intuitif, plus une base de réseau est « orthogonale », plus elle est à même de contenir un vecteur court. Cette heuristique se justifie par la notion de *déterminant d'un réseau* : étant donnée une base $v_1, \ldots, v_n$ d'un réseau, le déterminant de cette famille, c'est-à-dire le déterminant de la matrice ${}^t(v_1, \ldots, v_n)$ obtenue en plaçant les vecteurs lignes $v_i = (v_{i,1}, \ldots, v_{i,n})$ les uns au dessus des autres, est, au signe près, invariant par changement de base de $\mathbb{Z}^n$. En effet, soit $w_1, \ldots, w_n$ une autre base du même réseau. Il existe alors une matrice U à coefficients entiers, admettant un inverse

à coefficients entiers, tel que ${}^t(v_1,\dots,v_n) = \mathrm{U}\,{}^t(w_1,\dots,w_n)$. En passant aux déterminants, on a que $\det(\mathrm{U})$ vaut ± 1, donc que ${}^t(v_1,\dots,v_n)$ et ${}^t(w_1,\dots,w_n)$ ont même déterminant au signe près. Toujours au signe près, ce déterminant n'est autre que le volume du parallélépipède construit en s'appuyant sur les v_i. Les côtés de ce parallélépipède sont d'autant plus courts que ce parallélépipède est orthogonal.

Une base $v_1,\dots,v_n$ d'un réseau est dite *réduite* lorsque les orthogonalisés v_i^* des v_i par le procédé de Gram–Schmidt satisfont $\|v_i^*\|^2 \le 2\|v_{i+1}^*\|^2$ pour tout i entre 1 et $n-1$. En particulier, $\|v_1\|^2 = \|v_1^*\|^2 \ge 2^{i-1}\|v_i^*\|$. Les inégalités suivantes s'ensuivent pout tout vecteur non nul v du réseau :

$$\|v\| \ge \min_{1\le i\le n} \|v_i^*\| \ge \min_{1\le i\le n} 2^{-(i-1)/2}\|v_1\| \ge 2^{-(n-1)/2}\|v_1\|.$$

Autrement dit, le premier élément d'une base réduite est un vecteur court, au sens où pour tout v non nul du réseau on a $\|v_1\| \le 2^{(n-1)/2}\|v\|$.

20.3 L'algorithme LLL

Pour un réel x, on note $\lceil x \rfloor$ l'entier le plus proche de x (par défaut, l'entier immédiatement inférieur si x est un demi-entier). LLL est présenté dans l'Algorithme 20.2. Le reste du chapitre est essentiellement dédié à la preuve du théorème suivant, qui implique immédiatement le Théorème 20.1.

Théorème 20.2 L'Algorithme 20.2 est correct et a pour coût binaire

$$\tilde{\mathrm{O}}(n^5 \log^2 \mathrm{A}),$$

où $\mathrm{A} = \max_{1\le i\le n}\|v_i\|$.

Pour commencer, illustrons l'algorithme sur l'exemple donné par les deux vecteurs $v_1 = (6,4)$ et $v_2 = (8,4)$ de $\mathbb{Z}^2$. Après les étapes (1) et (2), on a $w_1^\star = v_1$, $w_2^\star = v_2 - \frac{16}{13}v_1 = (\frac{8}{13}, -\frac{12}{13})$, et

$$\mathrm{N} = \begin{pmatrix} 1 & 0 \\ \frac{16}{13} & 1 \end{pmatrix},$$

de sorte que $\mathrm{N}\,{}^t(w_1^\star, w_2^\star) = {}^t(w_1, w_2)$. Comme $\lceil \frac{16}{13} \rfloor = 1$, la transformation, dite de *réduction*, de l'étape (3.a) revient à multiplier N et ${}^t(w_1, w_2)$ à gauche par la matrice $\left(\begin{smallmatrix} 1 & 0 \\ -1 & 1 \end{smallmatrix}\right)$. Les vecteurs w_1 et w_2 deviennent $(6,4)$ et $(2,0)$, et la nouvelle valeur pour N est

$$\mathrm{N} = \begin{pmatrix} 1 & 0 \\ \frac{3}{13} & 1 \end{pmatrix}.$$

On a la relation

$$\mathrm{N}\begin{pmatrix} 6 & 4 \\ \frac{8}{13} & -\frac{12}{13} \end{pmatrix} = \begin{pmatrix} 6 & 4 \\ 2 & 0 \end{pmatrix},$$

qui montre que $w_1^\star, w_2^\star$ reste l'orthogonalisé de w_1, w_2.

En arrivant à l'étape (3.b), les normes à considérer vérifient $\|w_1^\star\|^2 = 52 > 2\|w_2^\star\|^2 = 32/13$, ce qui provoque l'*échange* des valeurs de w_1 et w_2. Nous pouvons vérifier sur

Entrée Des vecteurs $v_1,\dots,v_n$ de $\mathbb{Z}^n$ engendrant un réseau L.
Sortie Une base réduite $w_1,\dots,w_n$ de L.

1. Initialiser w_i à v_i pour chaque i de 1 à n.
2. Initialiser $w_1^\star,\dots,w_n^\star$ et $\mathrm{N} = (\nu_{i,j})$ avec les orthogonalisés de Gram–Schmidt $w_1^*,\dots,w_n^*$ de $w_1,\dots,w_n$ et la matrice M des $\mu_{i,j}$, de sorte que $\mathrm{N}\,{}^t(w_1^\star,\dots,w_n^\star) = {}^t(w_1,\dots,w_n)$.
3. Pour i à partir de 2, tant que $i \leq n$, faire :
   a. pour j de $i-1$ à 1, remplacer w_i par $w_i - \lceil \nu_{i,j} \rfloor w_j$ et remplacer $\nu_{i,k}$ par $\nu_{i,k} - \lceil \nu_{i,j} \rfloor \nu_{j,k}$ pour $1 \leq k \leq j$;
   b. si $i \geq 2$ et si $\|w_{i-1}^\star\|^2 > 2\|w_i^\star\|^2$,
      i. poser $\alpha = \nu_{i,i-1}$, et calculer $\beta = \nu_{i,i-1}\|w_{i-1}^\star\|^2/(\|w_i^\star\|^2 + \nu_{i,i-1}^2\|w_{i-1}^\star\|^2)$,
      ii. remplacer simultanément $w_{i-1}^\star$ et $w_i^\star$ par $w_i^\star + \alpha w_{i-1}^\star$ et $w_{i-1}^\star - \beta(w_i^\star + \alpha w_{i-1}^\star)$,
      iii. permuter les valeurs de w_{i-1} et w_i, ainsi que les lignes $i-1$ et i de N,
      iv. pour tout $i-1 \leq k \leq n$, remplacer simultanément $\nu_{k,i-1}$ et $\nu_{k,i}$ par $\beta\nu_{k,i-1} + (1-\alpha\beta)\nu_{k,i}$ et $\nu_{k,i-1} - \alpha\nu_{k,i}$.
      v. décrémenter i,
   sinon incrémenter i.
4. Renvoyer $w_1,\dots,w_n$.

Algorithme 20.2 – Algorithme de réduction de réseau LLL.

cette exemple que les formules de l'étape (3.b) correspondent à la mise à jour de l'orthogonalisé de Gram–Schmidt de w_1, w_2, puisque l'on obtient à la fin de cette étape : $\beta = 3$, $w_2^\star = (0,4)$, et

$$\mathrm{N} = \begin{pmatrix} 1 & 0 \\ 3 & 1 \end{pmatrix}$$

de sorte que

$$\mathrm{N}\begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 6 & 4 \end{pmatrix}. \tag{20.1}$$

La valeur de i passe alors de 2 à 1. On repasse ensuite par (3.a) et (3.b) mais sans rien faire, et i revient à 2. Comme $\lceil 3 \rfloor = 3$, une nouvelle réduction est effectuée à la nouvelle étape (3.a) : on multiplie alors à gauche les deux membres de (20.1) par $\left(\begin{smallmatrix} 1 & 0 \\ -3 & 1 \end{smallmatrix}\right)$. On obtient de la sorte la relation suivante :

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}.$$

Comme $\|w_1^*\|^2 = 4 \leq 2\|w_2^*\|^2 = 32$, l'algorithme termine en renvoyant la base réduite $w_1 = (2,0)$ et $w_2 = (0,4)$.

20.4 Preuve de l'algorithme LLL

Nous commençons par prouver que l'algorithme est correct et termine bien. Ensuite nous prouvons qu'il effectue $O(n^4 \log A)$ opérations arithmétiques dans $\mathbb{Q}$. Enfin, nous analysons la taille des nombres rationnels intervenant dans les calculs, de sorte à déduire la borne sur sa complexité binaire donnée dans le Théorème 20.2.

Correction

À l'entrée de la boucle (3), $w_1^\star,\dots,w_n^\star$ est l'orthogonalisé de Gram–Schmidt de $w_1,\dots,w_n$ et N est la matrice M des $\mu_{i,j}$ qui satisfait $M\,{}^t(w_1^*,\dots,w_n^*) = {}^t(w_1,\dots,w_n)$. Observons d'abord que cette propriété est préservée à chaque sortie du corps de boucle.

Il est clair qu'il en est ainsi après la transformation de l'étape (3.a) puisque N reste triangulaire supérieure de diagonale 1 et satisfait

$$N\,{}^t(w_1^\star,\dots,w_n^\star) = {}^t(w_1,\dots,w_n). \tag{20.2}$$

Quant à l'étape (3.b), on veut montrer que l'orthogonalisation est mise à jour après l'échange des vecteurs w_{i-1} et w_i. Notons $u_1,\dots,u_n$ la base

$$w_1,\dots,w_{i-2},w_i,w_{i-1},w_{i+1},\dots,w_n$$

du réseau obtenue après cet échange, et $u_1^*,\dots,u_n^*$ son orthogonalisé. On a les égalités entre espaces vectoriels $\mathbb{Q}w_1\oplus\cdots\oplus\mathbb{Q}w_k = \mathbb{Q}u_1\oplus\cdots\oplus\mathbb{Q}u_k$, pour $k\in\{1,\dots,n\}\setminus\{i-1\}$. Ainsi, les vecteurs orthogonalisés u_k^* et w_k^* sont clairement égaux lorsque $k\neq i-1,i$. Pour $k=i-1$, on déduit de

$$w_i = w_i^* + \alpha w_{i-1}^* + \mu_{i,i-2}w_{i-2}^* + \cdots + \mu_{i,1}w_1^*$$

que $u_{i-1}^* = w_i^* + \alpha w_{i-1}^*$ et donc

$$u_{i-1} = u_{i-1}^* + \mu_{i,i-2}u_{i-2}^* + \cdots + \mu_{i,1}u_1^*.$$

Ensuite on prétend que $u_i^* = w_{i-1}^* - \beta(w_i^* + \alpha w_{i-1}^*)$, avec

$$\beta = \frac{\beta\|w_{i-1}^*\|^2}{\|w_i^*\|^2 + \beta^2\|w_{i-1}^*\|^2}.$$

Ce vecteur est orthogonal à $u_1^*,\dots,u_{i-2}^*$, et on vérifie que

$$(u_i^* \mid u_{i-1}^*) = (w_{i-1}^* \mid w_i^* + \alpha w_{i-1}^*) - \beta\|w_i^* + \alpha w_{i-1}^*\|^2 = \alpha\|w_{i-1}^*\|^2 - \beta(\|w_i^*\|^2 + \alpha^2\|w_{i-1}^*\|^2) \;=\; 0.$$

Puis, à partir de

$$w_{i-1} = w_{i-1}^* + \mu_{i-1,i-2}w_{i-2}^* + \cdots + \mu_{i-1,1}w_1^*$$

on obtient

$$u_i = u_i^* + \beta u_{i-1}^* + \mu_{i-1,i-2}u_{i-2}^* + \cdots + \mu_{i-1,1}u_1^*.$$

Pour $k\geq i+1$, on utilise

$$w_{i-1}^* = u_i^* + \beta u_{i-1}^*, \quad w_i^* = -\alpha u_i^* + (1-\alpha\beta)u_{i-1}^*,$$

de sorte que

$$w_k = w_k^* + \mu_{k,k-1} w_{k-1}^* + \cdots + \mu_{k,1} w_1^*$$

se récrit en

$$u_k = u_k^* + \mu_{k,i+1} u_{i+1}^* + (\mu_{k,i-1} - \alpha\mu_{k,i}) u_i^* + (\beta\mu_{k,i-1} + (1-\alpha\beta)\mu_{k,i}) u_{i-1}^* + \mu_{k,i-2} u_{i-2}^* + \cdots + \mu_{k,1} w_1^*.$$

Donc à la sortie de l'étape (3.b), les $w_i^\star$ sont bien les orthogonalisés des w_i et la relation (20.2) est conservée.

Un second invariant respecté par la même boucle (3) est la suite d'inégalités

$$\|w_1^*\|^2 \leq 2\|w_2^*\|^2, \ldots, \|w_{i-2}^*\|^2 \leq 2\|w_{i-1}^*\|^2.$$

Cet invariant est bien satisfait à l'entrée de la boucle (3), pour $i = 2$. Puis, la stratégie de l'étape (3.b) est d'incrémenter i seulement si la suite d'inégalités peut se prolonger par $\|w_{i-1}^*\|^2 \leq 2\|w_i^*\|^2$, ou au contraire de décrémenter i lorsque w_{i-1}^* est modifié. Par conséquent, si une exécution de l'algorithme sort de la boucle (3), la base renvoyée est bien réduite. Il nous reste donc prouver maintenant que l'algorithme termine.

Terminaison

Pour montrer la terminaison de algorithme, on commence par observer que seule l'étape (3.b) modifie les $w_i^\star$. On va montrer que l'échange et la mise à jour durant cette étape réduit $\|w_{i-1}^\star\|^2$, et on va associer à ces normes une quantité entière qui décroît. Le nombre d'échanges ne pourra donc être que fini.

Lors d'un passage dans l'étape (3.b) avec $i \geq 2$ et $\|w_{i-1}^*\|^2 > 2\|w_i^*\|^2$, notons à nouveau $u_1, \ldots, u_n$ la base du réseau obtenue après l'échange de w_{i-1} et w_i. La formule $u_{i-1}^* = w_i^* + \alpha w_{i-1}^*$ implique

$$\|u_{i-1}^*\|^2 = \|w_i^*\|^2 + \alpha^2\|w_{i-1}^*\|^2 < \|w_{i-1}^*\|^2/2 + (1/2)^2\|w_{i-1}^*\|^2 = \frac{3}{4}\|w_{i-1}^*\|^2, \tag{20.3}$$

en utilisant le fait que l'étape (3.a) a forcé la relation $\alpha = \mu_{i,i-1} \leq 1/2$. Aussi on vérifie que la formule $u_i^* = w_{i-1}^* - \beta(w_i^* + \alpha w_{i-1}^*)$ implique

$$\begin{aligned} \|u_i^*\|^2 &= \|w_{i-1}^*\|^2 + \beta^2\|w_i^* + \alpha w_{i-1}^*\|^2 - 2(w_{i-1}^* \mid \beta(w_i^* + \alpha w_{i-1}^*)) \\ &= (1 - \alpha\beta)\|w_{i-1}^*\|^2 < \|w_{i-1}^*\|^2. \end{aligned} \tag{20.4}$$

On introduit la *matrice de Gram* G_i des vecteurs $w_1, \ldots, w_i$ comme la matrice de taille $i \times i$ dont l'entrée (k,l) est le produit scalaire $(w_k \mid w_l)$. C'est donc une matrice de $\mathbb{Z}^{i\times i}$, qui s'exprime aussi comme le produit

$$\mathrm{G}_i = {}^t(w_1, \ldots, w_i)(w_1, \ldots, w_i).$$

Puisque le bloc M_i de taille $i \times i$ en haut à gauche dans la matrice M est encore triangulaire inférieur de diagonale 1, on a

$$d_i = \det(\mathrm{M}_i\, {}^t(w_1, \ldots, w_i^*)^t(\mathrm{M}_i\, {}^t(w_1^*, \ldots, w_i^*))) = \|w_1^*\|^2 \cdots \|w_i^*\|^2.$$

Le variant qui va s'avérer adéquat est le produit $\mathrm{D} = d_1 \cdots d_{n-1}$, de nouveau un entier strictement positif. Lors d'un échange à l'étape (3.b), nous venons de prouver

que d_{i-1} est divisé par au moins 4/3. Comme les autre d_j restent inchangés, alors D est divisé par au moins 4/3, tout en restant entier strictement positif. Il ne peut donc y avoir qu'un nombre fini d'échanges lors d'une exécution de l'algorithme.

Considérons les nombres e d'échanges en (3.b) et f de passages en (3.b) qui incrémentent i. En observant ces valeurs à chaque entrée de la boucle (3), on a l'invariant $i = 2-e+f$. Chaque passage dans la boucle (3) incrémente l'un ou l'autre de ces deux nombres. Mais comme le nombre e ne peut croître indéfiniment, le nombre f doit ultimement ne plus cesser de croître, et i avec, ce jusqu'à la valeur $i = n+1$ qui provoque la fin de la boucle. L'algorithme termine donc.

Complexité arithmétique

Au début de l'algorithme, le nombre D qui vient d'être introduit pour montrer la terminaison vaut

$$\mathrm{D}_{\text{initial}} = \|v_1^*\|^{2(n-1)}\|v_2^*\|^{2(n-2)}\cdots\|v_{n-1}^*\|^2.$$

Mais chaque v_i^* étant alors une certaine projection de l'entrée v_i, on a

$$\mathrm{D}_{\text{initial}} \le \|v_1\|^{2(n-1)}\|v_2\|^{2(n-2)}\cdots\|v_{n-1}\|^2 \le \mathrm{A}^{n(n-1)}.$$

Après e échanges en (3.b), on a l'encadrement $1 \le \mathrm{D} \le (3/4)^e \mathrm{D}_{\text{initial}}$, d'où la borne supérieure $e \le n(n-1)\log_{4/3}\mathrm{A}$.

Notons e_{final} et f_{final} les valeurs finales de e et f, en fin d'exécution. On vient de prouver la relation $e_{\text{final}} = \mathrm{O}(n^2 \log \mathrm{A})$. Par ailleurs l'invariant sur i donne $n+1 = 2 - e_{\text{final}} + f_{\text{final}}$, d'où $v_{\text{final}} = \mathrm{O}(n^2 \log \mathrm{A})$. Comme l'orthogonalisation initiale se fait en $\mathrm{O}(n^3)$ opérations arithmétiques dans $\mathbb{Q}$, et que chacune des étapes (3.a) et (3.b) se fait en $\mathrm{O}(n^2)$ telles opérations arithmétiques, la complexité arithmétique totale de l'algorithme LLL est

$$\mathrm{O}(n^3) + (e_{\text{final}} + v_{\text{final}})\mathrm{O}(n^2) = \mathrm{O}(n^4 \log \mathrm{A}).$$

Complexité binaire

Il nous reste à analyser la taille des entiers intervenant dans l'algorithme. Nous allons montrer qu'elle reste dominée par $\mathrm{O}(n \log \mathrm{A})$, ce qui achèvera la preuve du Théorème 20.2.

Coût de l'orthogonalisation

Commençons par borner la taille des rationnels intervenant lors du procédé d'orthogonalisation. Notons $\mathrm{K} = \mathrm{M}^{-1}$ et notons $\kappa_{i,j}$ les entrées de K. Il s'agit d'une matrice triangulaire inférieure avec des 1 sur la diagonale. Pour tout $1 \le i \le n$, on a

$$w_i^* = w_i + \kappa_{i,i-1} w_{i-1} + \cdots + \kappa_{i,1} w_1.$$

En prenant les produits scalaires par w_j pour $1 \le j \le i-1$ on obtient

$$0 = (w_i, w_j) + \kappa_{i,i-1}(w_{i-1}, w_j) + \cdots + \kappa_{i,1}(w_1, w_j),$$

ce qui donne la relation

$$\mathrm{G}_{i-1}\,{}^t(\kappa_{i,1}, \ldots, \kappa_{i,i-1}) + {}^t((w_i, w_1), \ldots, (w_i, w_{i-1})) = 0, \tag{20.5}$$

qui représente $(\kappa_{i,1},\dots,\kappa_{i,i-1})$ comme solution d'un système linéaire.

Pour analyser l'étape (2) de l'algorithme il suffit d'utiliser le fait que les normes $\|v_i\| = \|w_i\|$ sont au plus A. Le système linéaire (20.5) est inversible car le déterminant de G_{i-1} est $d_{i-1} \geq 1$. Par les formules de Cramer, on déduit déjà que $d_{i-1}\kappa_{i,j}$ est un entier, pour tout $1 \leq j \leq i$. Par conséquent $d_{i-1}w_i^* \in \mathbb{Z}^n$ et $\|w_i^*\| \geq 1/\sqrt{d_{i-1}}$. Ensuite on peut borner les $\mu_{i,j}$ comme suit :

$$|\mu_{i,j}| = \frac{(w_i, w_j^*)}{\|w_j^*\|^2} \leq \frac{\|w_i\|}{\|w_j^*\|} \leq \sqrt{d_{j-1}}\|w_i\| \leq A^i. \tag{20.6}$$

Il s'ensuit que la taille des entiers intervenant dans l'étape (2) est bornée par $O(n \log A)$. Le coût binaire de cette étape est donc $O(n^4 \log A)$.

Taille des entiers dans la boucle principale

Grâce à (20.3) et (20.4), nous savons que les normes des $\|w_i^*\|$ restent bornées par A tout au long de l'algorithme, et que $d_i \leq A^i$. Commençons par majorer les normes des w_i au long de l'exécution. À partir de

$$\|w_i\|^2 = \|w_i^*\|^2 + \mu_{i,i-1}^2\|w_{i-1}^*\|^2 + \cdots + \mu_{i,i-1}^2\|w_{i-1}^*\|^2$$

nous avons

$$\|w_i\| \leq \sqrt{i}A \max_{1\leq j\leq i} \mu_{i,j}.$$

Lorsqu'on entre dans l'étape (3.a), la borne (20.6) garantit que les entiers des $\mu_{i,j}$ sont au plus A^i et donc que

$$\|w_i\| \leq \sqrt{i}A^{i+1}.$$

Lorsque $|\nu_{i,j}| \geq 1/2$, on remplace $\nu_{i,k}$ par $\nu_{i,k} - \lceil \nu_{i,j} \rfloor \mu_{j,k}$ au cours de l'étape (3.a), alors $|\nu_{i,k}|$ croît au plus de

$$|\lceil \nu_{i,k} \rfloor \mu_{j,k}| \leq (\nu_{i,j} + \frac{1}{2})\frac{1}{2} \leq \nu_{i,j},$$

lorsque $k < j$, comme $|\nu_{j,l}| \leq 1/2$. Lorsque $j = k$, on a $|\nu_{i,j}| \leq 1/2$. Donc les entiers $\nu_{i,j}$ restent bornés par $(2A)^i$. Par conséquent, les tailles des entiers dans les étapes (3.a) restent en $O(n \log A)$. Lorsqu'on quitte l'étape (3.a), tous les $\nu_{i,j}$ sont au plus 1/2 lorsque $j < i$, et donc $\|w_i\| \leq \sqrt{i}A$.

Enfin, lors de l'étape (3.b), il est clair que les entiers restent aussi de taille $O(n \log A)$, ce qui achève l'étude de la complexité binaire de l'algorithme LLL.

Bornes sur la base réduite

En complément, nous donnons les bornes suivantes, qui nous seront utiles dans le prochain chapitre :

Proposition 20.3 La base $w_1,\dots,w_n$ renvoyée par l'Algorithme 20.2 satisfait la propriété suivante : $\|w_j\|^2 \leq 2^{i-1}\|w_i^*\|^2$, pour tout $1 \leq j \leq i \leq n$.

Démonstration. Comme $w_1, \ldots, w_n$ est une base réduite, on a $\|w_j^*\|^2 \leq 2^{i-j}\|w_i^*\|^2$. Par ailleurs, à la sortie de l'algorithme, les entrées sous la diagonale de N sont au plus 1/2 en valeur absolue. La relation (20.2) implique donc

$$\begin{aligned}\|w_j\|^2 &\leq \|w_j^*\|^2 + \frac{1}{4}\left(\|w_{j-1}^*\|^2 + \cdots + \|w_1^*\|^2\right) \\ &\leq \left(2^{i-j} + \frac{1}{4}\left(2^{i-j+1} + \cdots + 2^{i-1}\right)\right)\|w_i^*\|^2 \leq 2^{i-1}\|w_i^*\|^2. \qquad \blacksquare\end{aligned}$$

Notes

La théorie des réseaux remonte aux travaux d'Hermite [Pic05] et à l'algorithme de réduction de Lagrange pour la dimension deux [Lag73 ; Lag75]. L'algorithme LLL a été découvert par A. Lenstra, H. Lenstra et L. Lovász en 1982 [LLL82]. Le présent chapitre suit la version décrite dans le Chapitre 16 du livre de von zur Gathen et Gerhard [GG03]. La preuve que le calcul du plus court vecteur d'un réseau est NP-difficile est due à Ajtai en 1998 [Ajt98].

En pratique, il est conseillé de modifier l'Algorithme 20.2 de sorte à ne pas calculer les $w_i^\star$, mais seulement les $\|w_i^\star\|^2$. Aussi pour améliorer significativement les performances, il ne faut pas effectuer les calculs avec des nombres rationnels, car les calculs de pgcd nécessaires pour maintenir les numérateurs et dénominateurs premiers entre eux sont très coûteux. On préfère donc programmer une version qui sépare habilement numérateurs et dénominateurs, en utilisant le procédé d'orthogonalisation décrit par Erlingsson, Kaltofen et Musser [EKM96]. D'autres variantes, avec des meilleures bornes de complexité, ont été proposées par plusieurs auteurs, dont Kaltofen [Kal83], Schönhage [Sch84], Koy et Schnorr [KS01 ; Sch88a]. La qualité de la base obtenue par les divers procédés a été étudiée par Schnorr [Sch03 ; Sch87].

Pour améliorer davantage la complexité de l'algorithme LLL, il faut commencer par observer que les $\lceil \mu_{i,i-1} \rfloor$ dépendent juste des bits de poids fort de $\mu_{i,i-1}$. On peut donc utiliser une arithmétique numérique en virgule flottante pour accélérer les calculs, et conserver les matrices unimodulaires de changement de base plutôt que les bases intermédiaires. Toute la difficulté pratique repose sur le choix de bonnes stratégies pour bénéficier au maximum de la double précision IEEE 754, native dans de nombreux processeurs. Pour bien faire fonctionner cette technique il faut aussi relâcher la condition de réduction. Ces idées ont été introduites par Schnorr en 1988 [Sch88a], puis améliorées avec Euchner [SE94] : elles ont conduit à la borne de complexité $O(n^4(n + \log A)^2 \log A)$, et à une implantation très efficace. Puis, dans cette veine, l'algorithme L^2, conçu par Nguyen et Stéhlé [NS05 ; NS09], a atteint la borne $(n^3(n + \log A)\log A)\tilde{O}(n))$ en utilisant une précision qui dépend juste linéairement de n. Maintenant, le meilleur algorithme connu, appelé L^1 est dû à Novocin, Stéhlé et Villard en 2011 [NSV11], et a pour coût $O(n^{5+\epsilon}\log A + n^{\theta+1+\epsilon}(\log A)^{1+\epsilon})$, où $\theta \leq 3$ est un exposant réalisable de l'algèbre linéaire (au sens du Chapitre 8), et $\epsilon > 0$ est une constante réelle qui peut être fixée arbitrairement petite.

Bibliographie

Ajt98 Ajtai, Miklós (1998). « The shortest vector problem in L2 is NP-hard for randomized reductions ». In : *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*. STOC '98. Extended abstract. New York, NY, USA : ACM, p. 10–19.

EKM96 Erlingsson, Úlfar, Erich Kaltofen et David Musser (1996). « Generic Gram–Schmidt orthogonalization by exact division ». In : *Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation*. ISSAC'96. New York, NY, USA : ACM, p. 275–282.

GG03 Gathen, Joachim von zur et Jürgen Gerhard (2003). *Modern computer algebra*. 2e éd. Cambridge University Press.

Kal83 Kaltofen, Erich (1983). « On the complexity of finding short vectors in integer lattices ». In : *Computer Algebra : EUROCAL'83, European Computer Algebra Conference London, England, March 28–30, 1983 Proceedings*. Éd. par J. A. Hulzen. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 236–244.

KS01 Koy, Henrik et Claus Peter Schnorr (2001). « Segment LLL-Reduction of Lattice Bases ». In : *Cryptography and Lattices : International Conference, CaLC 2001 Providence, RI, USA, March 29–30, 2001 Revised Papers*. Éd. par Joseph H. Silverman. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 67–80.

Lag73 Lagrange, Joseph Louis, comte de (1773). « Recherches d'arithmétique ». In : *Nouveaux Mémoires de l'Académie Royale des Sciences et Belles-Lettres de Berlin*, p. 265–312.

Lag75 — (1775). « Recherches d'arithmétique ». In : *Nouveaux Mémoires de l'Académie Royale des Sciences et Belles-Lettres de Berlin*, p. 323–356.

LLL82 Lenstra, A. K., H. W. Lenstra Jr. et L. Lovász (1982). « Factoring polynomials with rational coefficients ». In : *Mathematische Annalen*, vol. 261, n°4, p. 515–534.

NS05 Nguyên, Phong Q. et Damien Stehlé (2005). « Floating-point LLL revisited ». In : *Proceedings of Advances in Cryptology, EUROCRYPT 2005 : 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005*. Éd. par Ronald Cramer. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 215–233.

NS09 Nguyen, Phong Q. et Damien Stehlé (2009). « An LLL algorithm with quadratic complexity ». In : *SIAM Journal on Computing*, vol. 39, n°3, p. 874–903.

NSV11 Novocin, Andrew, Damien Stehlé et Gilles Villard (2011). « An LLL-reduction algorithm with quasi-linear time complexity ». In : *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*. STOC '11. Extended abstract. New York, NY, USA : ACM, p. 403–412.

Pic05 Picard, Émile, éd. (1905). *Œuvres de Charles Hermite*. Gauthier-Villars.

Sch03 Schnorr, Claus Peter (2003). « Lattice reduction by random sampling and birthday methods ». In : *STACS 2003 : 20th Annual Symposium on Theoretical Aspects of Computer Science Berlin, Germany, February 27 – March 1,*

2003 Proceedings. Éd. par Helmut Alt et Michel Habib. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 145–156.

Sch84 Schönhage, Arnold (1984). « Factorization of univariate integer polynomials by diophantine approximation and an improved basis reduction algorithm ». In : *Automata, Languages and Programming : 11th Colloquium Antwerp, Belgium, July 16–20, 1984*. Éd. par Jan Paredaens. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 436–447.

Sch87 Schnorr, C. P. (1987). « A hierarchy of polynomial time lattice basis reduction algorithms ». In : *Theoretical Computer Science*, vol. 53, n°2-3, p. 201–224.

Sch88a — (1988). « A more efficient algorithm for lattice basis reduction ». In : *Journal of Algorithms*, vol. 9, n°1, p. 47–62.

SE94 Schnorr, C. P. et M. Euchner (1994). « Lattice basis reduction : improved practical algorithms and solving subset sum problems ». In : *Mathematical Programming*, vol. 66, n°1, p. 181–199.

21. Factorisation des polynômes à une variable sur les rationnels

Résumé

La factorisation des polynômes à coefficients des nombres rationnels repose sur trois résultats fondamentaux : une borne sur la taille des coefficients des facteurs, l'algorithme de Newton–Hensel, et l'algorithme de réduction LLL. L'idée principale consiste à calculer la factorisation irréductible à coefficients p-adiques à une précision suffisante, pour un nombre p bien choisi. Ensuite il reste à deviner comment les facteurs p-adiques se recombinent en les facteurs rationnels.

Tout au long de ce chapitre F désigne le polynôme de degré n de $\mathbb{Q}[X]$ que nous souhaitons factoriser. Son coefficient de tête, c'est-à-dire de degré n, est noté $\mathrm{ct}(F)$.

Le *contenu* d'un polynôme de $\mathbb{Z}[X]$ est défini comme le pgcd de ses coefficients, et sa *partie primitive* correspond au quotient par son contenu. Rappelons le résultat classique général d'arithmétique suivant, que nous admettrons :

Proposition 21.1 — Lemme de Gauss sur les contenus. Soit $\mathbb{A}$ est un anneau factoriel, le contenu du produit de deux polynômes de $\mathbb{A}[X]$ est égal au produit de leurs contenus.

Théorème 21.2 — Lemme de Gauss sur les polynômes irréductibles. Soient $\mathbb{A}$ un anneau factoriel, et $\mathbb{K}$ son corps de fractions. Un polynôme non constant de $\mathbb{A}[X]$ est irréductible si, et seulement, s'il est primitif et irréductible dans $\mathbb{K}[X]$.

Démonstration. Soit F un polynôme non constant de $\mathbb{A}[X]$. Il est nécessairement primitif. S'il se factorise dans $\mathbb{K}[X]$, alors il existe des polynômes F_1 et F_2 primitifs de $\mathbb{A}[X]$, et un élément c de $\mathbb{A}$, tels que $cF = F_1F_2$. La proposition précédente implique

que c est une unité. Cela prouve le sens direct de l'énoncé. La réciproque est quant à elle immédiate. ■

Ici, la conséquence pratique de ce théorème est de réduire la factorisation de $\mathbb{Z}[X]$ à celle d'un contenu dans $\mathbb{Z}$ et celle d'un polynôme primitif dans $\mathbb{Q}[X]$.

La factorisation dans $\mathbb{Z}[X]$ semble donc mathématiquement proche de celle dans $\mathbb{Q}[X]$. Seulement, à l'heure actuelle, du point de vue de la complexité, nous ne connaissons toujours pas d'algorithme pour factoriser les entiers en temps polynomial. Nous n'aborderons pas cette question dans cet ouvrage. Néanmoins, une fois tout risque de confusion maintenant levé, il sera plus pratique de considerer que le polynôme à factoriser est à coefficients entiers et primitif. Les factorisations dans $\mathbb{Z}[X]$ et $\mathbb{Q}[X]$ coïncident alors. Dans la section suivante nous présentons un premier algorithme très simple de coût exponentiel qui conduit à une borne raisonnable sur les tailles des coefficients des facteurs irréductibles, que nous améliorerons ensuite.

21.1 Bornes sur les coefficients des facteurs

Dans cette section $F(X) = f_n X^n + f_{n-1} X^{n-1} + \cdots + f_1 X + f_0$ représente un polynôme primitif de $\mathbb{Z}[X]$ de degré n.

Borne naive

Si $G \in \mathbb{Z}[X]$ est un *facteur propre* de F, c'est-à-dire non constant et primitif, de degré au plus $m \leq n-1$, alors, pour tout entier $i \in \{0,\dots,m\}$, nécessairement $G(i)$ divise $F(i)$. Cette simple remarque fournit un algorithme pour trouver un tel facteur G ou bien montrer qu'il n'en existe pas, et prouver ainsi que F est irréductible :

1. Évaluer F en tous les points de $\{0,\dots,m\}$.
2. Énumérer tous les vecteurs $(b_0,\dots,b_m)$ de $\mathbb{Z}^m$ tels que b_i divise $F(i)$ pour tout $i \in \{0,\dots,m\}$.
3. Pour chaque tel vecteur $(b_0,\dots,b_m)$, interpoler le polynôme G de degré au plus m tel que $G(i) = b_i$ pour tout $i \in \{0,\dots,m\}$. Si G est non constant dans $\mathbb{Z}[X]$, et divise F, alors c'est un facteur propre de F.

Le coût de cette méthode dans le pire des cas est bien exponentiel. Néanmoins elle conduit à un premier résultat concernant la taille des coefficients des facteurs de F. Notons $\|F\|_\infty$ la quantité $\max_{i\in\{0,\dots,n\}} |f_i|$.

Proposition 21.3 Soit F un polynôme primitif de $\mathbb{Z}[X]$ de degré n, et soit G un facteur propre de F de degré $m \leq n-1$. Alors $\|G\|_\infty \leq (m+1)(n+1)m^{n+2m}\|F\|_\infty$.

Démonstration. À partir de la formule d'interpolation de Lagrange nous avons

$$G(X) = \sum_{i=0}^{m} G(i) \prod_{j=0, j\neq i}^{m} \frac{X-j}{i-j},$$

et déduisons la majoration suivante :

$$\|G\|_\infty \le \left(\max_{i\in\{0,\dots,m\}} |G(i)|\right) \left\| \sum_{i=0}^{m} \prod_{j=0, j\neq i}^{m} (X+j) \right\|_\infty \le (m+1)m^{2m} \max_{i\in\{0,\dots,m\}} |G(i)|.$$

La conclusion provient de

$$\max_{i\in\{0,\dots,m\}} |G(i)| \le \max_{i\in\{0,\dots,m\}} |F(i)| \le (n+1)m^n \|F\|_\infty. \qquad \blacksquare$$

La taille des coefficients de G est donc au plus la taille de ceux de F plus $O(n \log n)$.

Exemple 21.1 Avec $F = X^{105} - 1 = (X-1)(X^2+X+1)(X^4+X^3+X^2+X+1)(X^6+X^5+X^4+X^3+X^2+X+1)(X^8-X^7+X^5-X^4+X^3-X+1)(X^{12}-X^{11}+X^9-X^8+X^6-X^4+X^3-X+1)(X^{24}-X^{23}+X^{19}-X^{18}+X^{17}-X^{16}+X^{14}-X^{13}+X^{12}-X^{11}+X^{10}-X^8+X^7-X^6+X^5-X+1)(X^{48}+X^{47}+X^{46}-X^{43}-X^{42}-2X^{41}-X^{40}-X^{39}+X^{36}+X^{35}+X^{34}+X^{33}+X^{32}+X^{31}-X^{28}-X^{26}-X^{24}-X^{22}-X^{20}+X^{17}+X^{16}+X^{15}+X^{14}+X^{13}+X^{12}-X^9-X^8-2X^7-X^6-X^5+X^2+X+1)$, la taille des coefficients du dernier facteur dépasse légèrement celle de F.

Borne de Landau–Mignotte

Dans la suite de ce chapitre la borne donnée dans la Proposition 21.3 sera suffisante pour obtenir un algorithme de factorisation en temps polynomial. En pratique il est néanmoins utile d'avoir des bornes plus précises. Pour ce faire, nous utiliserons la norme $\|.\|_2$, définie par

$$\|F\|_2 = \left(|f_n|^2 + \cdots + |f_0|^2\right)^{1/2}.$$

L'idée pour borner plus finement la taille des coefficients d'un facteur de F est de considérer la *mesure de Mahler* de $F(X) = f_n \prod_{i=1}^{n} (X-\varphi_i) \in \mathbb{C}[X] \setminus \{0\}$, notée M(F), et définie par

$$M(F) = |f_n| \prod_{i=1}^{n} \max(1, |\varphi_i|).$$

Ici les racines φ_i ne sont pas nécessairement distinctes puisque F n'est pas supposé sans carré. Cette fonction M est multiplicative : si F = GH, alors nous avons visiblement M(F) = M(G)M(H).

Dans un premier temps nous allons montrer $M(F) \le \|F\|_2$. Notons que cette borne est immédiate si toutes les racines de F sont de module au plus 1, puisque dans ce cas on a $M(F) = ct(F) \le \|F\|_2$. Pour se ramener à ce cas nous utilisons le polynôme auxiliaire

$$H(X) = ct(F)\bar{\varphi}_{m+1} \cdots \bar{\varphi}_n (X-\varphi_1)\cdots(X-\varphi_m)(X-\bar{\varphi}_{m+1}^{-1})\cdots(X-\bar{\varphi}_n^{-1}),$$

où les racines $\varphi_1, \dots, \varphi_n$ de F sont ordonnées par modules croissants, et où $m \in \{0, \dots, n\}$ est défini par $0 < |\varphi_1| \le \cdots \le |\varphi_m| < 1 \le |\varphi_{m+1}| \le \cdots \le |\varphi_n|$. Ce polynôme H a toutes ses racines de modules au plus 1, et donc $M(H) = ct(H) \le \|H\|_2$. De plus, la construction de H est faite pour avoir M(H) = M(F).

Proposition 21.4 Pour tout polynôme F de $\mathbb{C}[X]$, on a $M(F) \leq \|F\|_2$.

Démonstration. Avec ce que nous venons de voir à propos de H, il nous reste à prouver $\|F\|_2 = \|H\|_2$. Soit ω une racine primitive de l'unité d'ordre $n+1$ dans $\mathbb{C}$. La transformée de Fourier associée à ω,

$$F \mapsto (F(1), F(\omega), \dots, F(\omega^n)),$$

définit un changement de base dans l'espace des polynômes de degré au plus n. Le Lemme 2.7 nous garantit que ce changement de base est orthogonal, et que les vecteurs de la nouvelle base ont pour norme $1/\sqrt{n+1}$. Par conséquent on a

$$\|F\|_2^2 = \frac{1}{n+1} \sum_{i=0}^{n} |F(\omega^i)|^2.$$

En utilisant la formule similaire pour H, et en vérifiant que l'on a $|F(\omega^i)|^2 = |H(\omega^i)|^2$ pour tout i, on obtient bien $\|F\|_2 = \|H\|_2$. ■

Les formules classiques reliant les coefficients de F à ses racines mènent alors à :

Proposition 21.5 Si G est un facteur de degré m de F, alors

$$\|G\|_2 \leq 2^m M(G) \leq 2^m \left| \frac{\mathrm{ct}(G)}{\mathrm{ct}(F)} \right| \|F\|_2.$$

Démonstration. Si g_i est le coefficient de degré i de G et si $\varphi_1, \dots, \varphi_m$ sont les racines de G classées par ordre de croissant de module, alors on a $|g_i| \leq |g_m| \binom{m}{i} |\varphi_{i+1}| \cdots |\varphi_m| \leq \binom{m}{i} M(G)$. De l'inégalité $\sum_{i=0}^{m} \binom{m}{i}^2 \leq \left(\sum_{i=0}^{m} \binom{m}{i} \right)^2 = 2^{2m}$ nous déduisons $\|G\|_2 \leq 2^m M(G)$. La preuve s'achève donc en utilisant $M(G) \leq \left| \frac{g_m}{f_n} \right| M(F)$ et la proposition précédente. ■

Nous déduisons le théorème suivant souvent appelé *borne de Landau–Mignotte*.

Théorème 21.6 Si $G \in \mathbb{Z}[X]$ est un facteur de $F \in \mathbb{Z}[X]$ de degré m, alors

$$\|\mathrm{ct}(F) G / \mathrm{ct}(G)\|_\infty \leq \sqrt{n+1}\, 2^m \|F\|_\infty.$$

Démonstration. Il s'agit juste de combiner les inégalités $\|G\|_\infty \leq \|G\|_2$, et $\|F\|_2 \leq \sqrt{n+1} \|F\|_\infty$ avec la proposition précédente. ■

Borne de Mahler

En réutilisant le polynôme auxiliaire H de la sous-section précédente, nous pouvons aussi obtenir l'inégalité suivante due à Mahler, qui nous sera utile pour l'algorithme de van Hoeij présenté en Section 21.4.

Théorème 21.7 Pour tout $F \in \mathbb{C}[X]$ de degré n, on a $M(F') \leq nM(F)$.

Démonstration. Le polynôme H a toutes ses racines dans le disque unité ouvert. Par le classique *théorème de Gauss–Lucas* (rappelé dans le Lemme 21.9 ci-dessous), les racines de $H'(X)$ sont dans l'enveloppe convexe de celles de $H(X)$, et donc dans le disque unité ouvert. On a ainsi $M(H') = nM(H)$. Comme $M(H) = M(F)$, il suffit de prouver $M(F') \leq M(H')$.

Si $|x| \geq 1$ et $i \geq m+1$, alors on vérifie l'inégalité $|x - \varphi_i| \leq |\bar{\varphi}_i x - 1|$, en constatant qu'elle est équivalente à

$$(x - \varphi_i)(\bar{x} - \bar{\varphi}_i) \leq (\bar{\varphi}_i x - 1)(\varphi_i \bar{x} - 1),$$

qui se récrit en $0 \leq |\varphi_i|^2|x|^2 - |x|^2 - |\varphi_i|^2 + 1 = (|\varphi_i|^2 - 1)(|x|^2 - 1)$. Pour tout $|x| \geq 1$, on a donc $|F(x)| \leq |H(x)|$. En d'autres termes, si $u \in \mathbb{C}$ est de module $|u| > 1$, alors les racines de $F(X) - uH(X)$ sont toutes dans le disque unité ouvert. Par le théorème de Gauss–Lucas, les racines de $F'(X) - uH'(X)$ sont donc aussi dans le disque unité ouvert. Par conséquent, pour tout $x \in \mathbb{C}$ n'annulant pas H', si $u = F'(x)/H'(x)$ est de module strictement plus grand que 1, alors x est un zéro de $F'(X) - uH'(X)$ et donc dans le disque unité ouvert. Autrement dit, on vient de prouver $|F'(x)| \leq |H'(x)|$ pour tout $|x| \geq 1$. Le lemme suivant assure alors que $M(F') \leq M(H')$. ■

Lemme 21.8 Si F et G sont deux polynômes de $\mathbb{C}[X]$ tels que $|F(x)| \leq |G(x)|$ pour tout x de module 1, alors $M(F) \leq M(G)$.

Démonstration. On considère le résultant $\mathrm{Res}(F(X), X^k - 1)$. La formule de Poisson vue dans le Théorème 6.5 conduit à l'identité

$$\mathrm{ct}(F)^k \prod_{F(x)=0} (x^k - 1) = (-1)^{kn} \prod_{x^k=1} F(x),$$

qui implique

$$|\mathrm{ct}(F)| \prod_{F(x)=0} |x^k - 1|^{1/k} = \prod_{x^k=1} |F(x)|^{1/k}.$$

En faisant de même pour G, et en utilisant l'hypothèse $|F(x)| \leq |G(x)|$, on déduit

$$|\mathrm{ct}(F)| \prod_{F(x)=0} |x^k - 1|^{1/k} \leq |\mathrm{ct}(G)| \prod_{G(x)=0} |x^k - 1|^{1/k}.$$

Si $|x| < 1$, alors $|x^k - 1|^{1/k}$ converge vers 1 lorsque k tend vers l'infini. Si $|x| > 1$ alors $|x^k - 1|^{1/k}$ converge vers $|x|$. Si ni F ni G n'ont de racine de module 1 alors on en déduit $M(F) \leq M(G)$.

Par hypothèse, les racines de G qui sont sur le cercle unité annulent aussi F. En divisant F et G par le polynôme annulateur unitaire de ces racines, par la multiplicativité de M, on se ramène au cas où G n'a pas de racine de module 1. Ensuite on prend α réel au voisinage de 1 de sorte que $F(\alpha X)$ n'admet aucune racine de module

1 lorsque $\alpha > 1$. Quitte à restreindre le voisinage pour α, il existe un voisinage réel pointé à gauche de 1 qui contient des valeurs β telles que $\beta F(\alpha x) \leq G(x)$ pour tout x de module 1. On a alors $M(\beta F(\alpha X)) \leq M(G)$. On peut enfin faire tendre α vers 1 puis β vers 1 dans cette dernière inégalité pour conclure la preuve du lemme. ■

Lemme 21.9 — Gauss–Lucas. Les racines de la dérivée d'un polynôme $F \in \mathbb{C}[X]$ non constant sont dans l'envelope convexe des racines de F.

Démonstration. Si $F(X) = \mathrm{ct}(F)\prod_{i=1}^{n}(X-\varphi_i)$, alors la dérivée logarithmique de F est

$$\frac{F'}{F} = \sum_{i=1}^{n} \frac{1}{X-\varphi_i}.$$

Soit ψ est une racine de F'. Le résultat est immédiat si ψ est aussi racine de F. Sinon on a

$$\sum_{i=1}^{n} \frac{1}{\psi-\varphi_i} = 0,$$

qui implique

$$\sum_{i=1}^{n} \frac{\bar{\psi}-\bar{\varphi}_i}{|\psi-\varphi_i|^2} = 0,$$

et donc

$$\psi \sum_{i=1}^{n} \frac{1}{|\psi-\varphi_i|^2} = \sum_{i=1}^{n} \frac{\varphi_i}{|\psi-\varphi_i|^2}.$$

Autrement dit, ψ est un barycentre des ϕ. ■

R La définition de la mesure de Mahler peut sembler *a priori* artificielle. En fait, une définition plus naturelle est la formule suivante :

$$\log M(F) = \frac{1}{2\pi}\int_0^{2\pi} \log|F(e^{i\theta})|\mathrm{d}\theta.$$

L'équivalence des deux définitions fait habituellement appel à des résultats classiques d'analyse complexe, mais on peut aussi y parvenir en adaptant les arguments utilisés dans la preuve du Lemme 21.8. En effet, nous avons vu que si F n'a pas de racine sur le cercle unité, alors le membre de gauche de l'identité

$$|\mathrm{ct}(F)| \prod_{F(x)=0} |x^k-1|^{1/k} = \prod_{x^k=1} |F(x)|^{1/k},$$

converge vers M(F). Par ailleurs, le logarithme du membre de droite vaut

$$\sum_{x^k=1} \frac{1}{k}\log|F(x)|$$

et converge donc vers

$$\frac{1}{2\pi}\int_0^{2\pi} \log|F(e^{i\theta})|d\theta$$

lorsque k tend vers l'infini. Enfin pour traiter le cas où F a des racines sur le cercle unité, il suffit de reprendre l'argument de déformation utilisé à la fin de la preuve du lemme.

Borne d'Hadamard

Afin de borner des résultants comme Res(F, F'), nous utilisons le résultat classique suivant :

Proposition 21.10 — Borne d'Hadamard. Si V est une matrice carrée de taille $n \times n$ à coefficients dans $\mathbb{R}$, alors $|\det(V)| \leq \|v_1\|_2 \cdots \|v_n\|_2$, où v_i représente la i-ième ligne de V.

Démonstration. Le procédé d'orthogonalisation de Gram–Schmidt, vu dans le chapitre précédent, appliqué aux vecteurs $v_1, \ldots, v_n$, fournit des vecteurs orthogonaux $v_1^*, \ldots, v_n^*$ et une matrice M triangulaire inférieure de diagonale unitaire tels que $\|v_i^*\|_2 \leq \|v_i\|_2$ pour tout i, et

$$M\begin{pmatrix} v_1^* \\ \vdots \\ v_n^* \end{pmatrix} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}.$$

Par conséquent le déterminant de V est égal à $\|v_1^*\|_2 \cdots \|v_n^*\|_2 \leq \|v_1\|_2 \cdots \|v_n\|_2$. ■

Corollaire 21.11 Si F et G sont deux polynômes de degrés respectifs n et m à coefficients réels, alors $|\mathrm{Res}(F, G)| \leq \|F\|_2^m \|G\|_2^n$.

Démonstration. Il suffit d'appliquer la proposition précédente à la matrice de Sylvester des polynômes F et G, définie au Chapitre 6. ■

21.2 Factorisation sans carré

Nous abordons maintenant la complexité binaire de la factorisation sans carré d'un polynôme primitif F de $\mathbb{Z}[X]$. La méthode présentée ici consiste à calculer la factorisation sans carré de F modulo p pour suffisamment de nombres premiers p, puis à utiliser la méthode des restes chinois pour reconstruire la factorisation sans carré de F dans $\mathbb{Z}[X]$. Nous commençons par étudier la complexité du calcul de nombres premiers.

Calcul de nombres premiers

Le premier ingrédient technique nécessaire est le calcul de nombres premiers plus petits qu'un entier k fixé. Le second ingrédient est un encadrement du produit des nombres premiers inférieurs à k.

Lemme 21.12 L'ensemble des nombres premiers inférieurs ou égaux à k peut se calculer en coût binaire $O(k \log^2 k) = \tilde{O}(k)$.

Démonstration. La borne de complexité découle de l'utilisation de la méthode classiquement appelée *crible d'Ératosthène*. On initialise un vecteur de booléens avec vrai de taille $k+1$: la i-ième entrée indiquera à la fin de l'algorithme si l'entier i est premier. On met les entrées 0 et 1 à faux. Le premier entier p dont l'entrée est vrai est 2, et on marque faux tous les multiples stricts de p à partir de p^2. On prend ensuite pour p l'entier suivant marqué vrai, en l'occurrence 3, et on marque faux tous les multiples stricts de p à partir de p^2. On prend ensuite pour p l'entier suivant marqué vrai, et on continue de la même façon. De la sorte, p est toujours premier. Il suffit de s'arrêter dès que $p > \sqrt{k}$. Le coût binaire de cette méthode est donc

$$O\left(\sum_{\substack{2 \le p \le \sqrt{k} \\ p \text{ premier}}} \frac{k}{p} \log k\right) = O\left((k \log k) \sum_{2 \le i \le k} \frac{1}{i}\right).$$

Cette dernière somme est majorée par

$$\sum_{2 \le i \le k} \frac{1}{i} \le \sum_{2 \le i \le k} \int_{i-1}^{i} \frac{1}{t} \mathrm{d}t \le \int_{1}^{k} \frac{1}{t} \mathrm{d}t = O(\log k),$$

ce qui nous donne le coût binaire annoncé $O(k \log^2 k)$. ∎

Lemme 21.13 Lorsque k est suffisamment grand on a

$$\left(\frac{3}{2}\right)^k \le \prod_{\substack{2 \le p \le k \\ p \text{ premier}}} p \le 4^{k-1}.$$

Démonstration. Commençons par la majoration. L'inégalité se vérifie directement pour $k = 2, 3$, puis nous entamons une récurrence sur k. Bien entendu, nous pouvons nous ramener à k impair de la forme $2m+1$, et supposer la borne valable pour m à la place de k.

On décompose le produit

$$\prod_{\substack{2 \le p \le 2m+1 \\ p \text{ premier}}} p \quad \text{en} \quad A = \prod_{\substack{2 \le p \le m \\ p \text{ premier}}} p \quad \text{et} \quad B = \prod_{\substack{m+1 \le p \le 2m+1 \\ p \text{ premier}}} p.$$

Par récurrence on a justement $A \le 4^{m-1}$. Ensuite on remarque que B divise le nombre combinatoire $\binom{2m+1}{m}$. Puis pour borner ce nombre on fait appel à la formule du binôme de Newton

$$(1+1)^{2m+1} = \sum_{i=0}^{2m+1} \binom{2m+1}{i},$$

qui donne $B \le \binom{2m+1}{m} \le 2^{2m+1}$, et donc $AB \le 4^{2m-1/2} \le 4^{k-1}$.

Pour la minoration, on commence par observer que la multiplicité de p dans $i \in \{m+1,\dots,2m\}$ est au plus $\log_p(2m)$, et comme il n'y a qu'au plus $1+\log_p m$ multiples de p dans $m+1,\dots,2m$, alors la multiplicité de p dans $C_n = (2m)(2m-1)\cdots(m+1)/m!$ est au plus $(1+\log_p m)\log_p(2m)$. De plus, si $\sqrt{2m} < p \le 2m$, alors cette multiplicité est au plus 1 : en effet la multiplicité de p dans $i \le 2m$ est au plus 1, le nombre de multiples de p dans $1,\dots,m$ est $\lfloor \frac{m}{p} \rfloor$ et le nombre de multiples de p dans $m+1,\dots,2m$ est au plus $1+\lfloor \frac{m}{p} \rfloor$. On déduit

$$C_n \le \left(\prod_{\substack{2\le p\le \sqrt{2m} \\ p \text{ premier}}} p^{(1+\log_p m)\log_p(2m)} \right) \left(\prod_{\substack{\sqrt{2m}<p\le 2m \\ p \text{ premier}}} p \right) \le (2m)^{(1+\log_2 m)\sqrt{2m}} \prod_{\substack{2\le p\le 2m \\ p \text{ premier}}} p.$$

Comme $C_n = \binom{2m}{m} \ge \binom{2m}{i}$ pour tout $0 \le i \le 2m$, l'identité

$$(1+1)^{2m} = \sum_{i=0}^{2m} \binom{2m}{i} \le 2 + \sum_{i=1}^{2m-1} \binom{2m}{i} \le 2m \binom{2m}{m},$$

implique alors $C_n \ge 4^m/(2m)$. Donc on a

$$\prod_{\substack{2\le p\le 2m \\ p \text{ premier}}} p \ge \frac{2^{2m}}{(2m)^{1+(\log_2(2m))\sqrt{2m}}},$$

et la conclusion provient de

$$\prod_{\substack{2\le p\le k \\ p \text{ premier}}} p \ge \frac{2^k}{(k+1)^{1+(\log_2(k+1))\sqrt{k+1}}} \ge \left(\frac{3}{2}\right)^k,$$

pour k suffisamment grand. La constante $\frac{3}{2}$ est arbitraire ici : n'importe quel nombre inférieur strict à 2 convient. ■

Méthode par restes chinois

Nous avons défini au Chapitre 18 la factorisation sans carré de F, notée fsc(F), comme l'ensemble des paires (Q, m), où Q est le produit des facteurs sans carré primitifs de F de multiplicité $m \ge 1$. Un entier p sera dit *compatible* avec la factorisation sans carré de F lorsque la factorisation sans carré de F mod p coïncide avec $\{(Q \bmod p, m) \mid (Q, m) \in \text{fsc}(F)\}$. Nous commençons par deux lemmes techniques.

Lemme 21.14 Soit F un polynôme de $\mathbb{Z}[X]$. Un nombre premier p divise $\text{Res}(F, F')$ si et seulement s'il divise $\text{ct}(F)$ ou si $\text{Res}(F \bmod p, F' \bmod p) = 0 \bmod p$.

Démonstration. Nous avons vu au Chapitre 6, que si A et B sont deux polynômes en la variable X de degrés respectifs m et n,

$$A(X) = a_m X^m + \cdots + a_1 X + a_0 \quad \text{et} \quad B(X) = b_n X^n + \cdots + b_1 X + b_0,$$

le résultant de A et B peut être calculé comme le déterminant de la matrice de Sylvester, carrée de taille $m+n$ suivante

$$\mathrm{Syl}(A,B) = \begin{pmatrix} a_m & \dots & a_1 & a_0 & & \\ & \ddots & & & \ddots & \\ & & a_m & a_{m-1} & \dots & a_0 \\ b_n & \dots & b_1 & b_0 & & \\ & \ddots & & & \ddots & \\ & & b_n & b_{n-1} & \dots & b_0 \end{pmatrix}$$

En considérant $\mathrm{Res}(F,F')$ comme le déterminant de la matrice de Sylvester de F et F′, nous vérifions que si $\mathrm{ct}(F) \bmod p \neq 0$, alors $\mathrm{Res}(F,F') \bmod p$ est proportionnel sur $\mathbb{F}_p$ à $\mathrm{Res}(F \bmod p, F' \bmod p)$. Par ailleurs, on observe aisément que $\mathrm{Res}(F,F')$ s'annule modulo p lorsque $\mathrm{ct}(F)$ s'annule modulo p. ■

Lemme 21.15 Soit p un nombre premier supérieur ou égal à $n+1$. Notons $G = \mathrm{pgcd}(F,F')$ et $H = F/G$ la partie sans carré de F. Les conditions suivantes sont équivalentes :
- p est compatible avec la factorisation sans carré de F ;
- p ne divise pas $\mathrm{Res}(H,H')$;
- p ne divise pas $\mathrm{ct}(F)$ et $\deg(\mathrm{pgcd}(F \bmod p, F' \bmod p)) = \deg G$.

Démonstration. Commençons par remarquer que l'hypothèse $p \geq n+1$ assure que les factorisations séparable et sans carré de F mod p coïncident.

La partie sans carré H de F n'est autre que le produit des facteurs sans carré de F, c'est-à-dire $H = \prod_{(Q,m)\in \mathrm{fsc}(F)} Q$. À partir de la dérivation $H' = \sum_{(Q,m)\in\mathrm{fsc}(F)} Q'(H/Q)$, et des propriétés multiplicatives du résultant énoncées dans le Théorème 6.5 et le Corollaire 6.6, on obtient :

$$\begin{aligned} \mathrm{Res}(H,H') &= \prod_{(Q,m)\in\mathrm{fsc}(F)} \mathrm{Res}(Q,H') \\ &= \prod_{(Q,m)\in\mathrm{fsc}(F)} \mathrm{ct}(Q)^* \,\mathrm{Res}(Q,Q'(H/Q)) \\ &= \pm \prod_{(Q,m)\in\mathrm{fsc}(F)} \mathrm{ct}(Q)^* \,\mathrm{Res}(Q,Q')\,\mathrm{Res}(Q,H/Q), \end{aligned}$$

où $\mathrm{ct}(Q)^*$ représente un puissance de $\mathrm{ct}(Q)$ qu'il n'est pas nécessaire d'expliciter ici. Le fait que p soit compatible signifie d'une part que les facteurs Q mod p sont séparables pour tout (Q,m) de $\mathrm{fsc}(F)$, et que d'autre part ces Q mod p sont premiers entre eux

deux à deux. Par la propriété de spécialisation du résultant vue dans le Lemme 21.14, cela est bien équivalent au fait que p ne divise pas $\mathrm{Res}(\mathrm{H}, \mathrm{H}')$.

Si p ne divise pas $\mathrm{Res}(\mathrm{H}, \mathrm{H}')$, alors $\mathrm{ct}(\mathrm{H}) \bmod p$ et $\mathrm{Res}(\mathrm{H} \bmod p, \mathrm{H}' \bmod p)$ sont non nuls par la propriété de spécialisation du résultant, ce qui signifie que $\mathrm{H} \bmod p$ est séparable : c'est donc la partie sans carré de $\mathrm{F} \bmod p$. On peut voir que les degrés de $\mathrm{pgcd}(\mathrm{F} \bmod p, \mathrm{F}' \bmod p)$ et G coïncident bien grâce à la formule suivante et au fait que p ne divise aucune multiplicité m :

$$\mathrm{pgcd}(\mathrm{F} \bmod p, \mathrm{F}' \bmod p) = \mathrm{pgcd}\left(\mathrm{H} \bmod p, \sum_{(\mathrm{Q},m)\in \mathrm{fsc}(\mathrm{F})} m(\mathrm{Q}' \bmod p)(\mathrm{H}/\mathrm{Q} \bmod p)\right) \prod_{(\mathrm{Q},m)\in \mathrm{fsc}(\mathrm{F})} (\mathrm{Q} \bmod p)^{m-1}.$$

Inversement, supposons que p divise $\mathrm{Res}(\mathrm{H}, \mathrm{H}')$. Si p ne divise pas $\mathrm{ct}(\mathrm{F})$, alors il ne divise pas $\mathrm{ct}(\mathrm{H})$ et donc $\mathrm{Res}(\mathrm{H} \bmod p, \mathrm{H}' \bmod p)$ est nul, ce qui implique que

$$\mathrm{pgcd}(\mathrm{H} \bmod p, \mathrm{H}' \bmod p) = \mathrm{pgcd}\left(\mathrm{H} \bmod p, \sum_{(\mathrm{Q},m)\in \mathrm{fsc}(\mathrm{F})} (\mathrm{Q}' \bmod p)(\mathrm{H}/\mathrm{Q} \bmod p)\right)$$

est non constant. Comme ce dernier pgcd coïncide avec

$$\mathrm{pgcd}\left(\mathrm{H} \bmod p, \sum_{(\mathrm{Q},m)\in \mathrm{fsc}(\mathrm{F})} m(\mathrm{Q}' \bmod p)(\mathrm{H}/\mathrm{Q} \bmod p)\right),$$

il s'ensuit que $\mathrm{pgcd}(\mathrm{F} \bmod p, \mathrm{F}' \bmod p)$ est de degré strictement plus grand que $\deg \mathrm{G}$, ce qui conclut la preuve. ■

La stratégie pour obtenir $\mathrm{fsc}(\mathrm{F})$ est la suivante : on calcule $\mathrm{fsc}(\mathrm{F} \bmod p)$ pour suffisamment de valeurs de nombres premiers $p \geq n+1$ de sorte qu'on puisse d'une part déterminer ceux qui sont compatibles, et d'autre part qu'il y ait suffisamment de compatibles pour pouvoir reconstruire les facteurs par la méthode des restes chinois. Les détails sont donnés dans l'Algorithme 21.1.

Proposition 21.16 L'Algorithme 21.1 est correct, de coût binaire $\tilde{\mathrm{O}}(n^2 \log \|\mathrm{F}\|_\infty)$.

Démonstration. Par la Proposition 6.2, le degré d de $\mathrm{G} = \mathrm{pgcd}(\mathrm{F}, \mathrm{F}')$ correspond à la dimension du noyau de la matrice de Sylvester de F et F'. Il existe donc un mineur M non nul de taille $2n-1-d$ de cette matrice tel que, si p ne divise ni M ni $\mathrm{ct}(\mathrm{F})$, alors le degré de $\mathrm{pgcd}(\mathrm{F} \bmod p, \mathrm{F}' \bmod p)$ est égal à d. Par le Lemme 21.15, un tel $p \geq n+1$ est alors compatible avec la factorisation sans carré de F.

Il nous faut maintenant prouver que l'ensemble E déterminé par l'algorithme permet bien de reconstruire les facteurs sans carré de F. D'une part, la borne d'Hadamard vue en Proposition 21.10 nous donne

$$|\mathrm{ct}(\mathrm{F})\mathrm{M}| \leq n^n \|\mathrm{F}\|_2^{2n} \leq n^{2n} \|\mathrm{F}\|_\infty^{2n}.$$

Entrée $F \in \mathbb{Z}[X]$, séparable et primitif de degré n.
Sortie La factorisation sans carré de F.

1. Calculer le premier entier k tel que
$$(3/2)^k > 2 \cdot 8^n \sqrt{n+1}\, n^{2n} \|F\|_\infty^{2n+1}.$$
2. Calculer tous les nombres premiers $\leq k$.
3. Pour tout nombre premier p entre $n+1$ et k, calculer fsc(F mod p).
4. Sélectionner l'ensemble E des nombres premiers p entre $n+1$ et k tels que la partie sans carré de F mod p soit de degré maximal.
5. Notons $Q_{p,m}$ le facteur sans carré de multiplicité m de F mod p.
Pour chaque multiplicité m, calculer l'unique polynôme Q_m de $\mathbb{Z}[X]$ satisfaisant $Q_m = \mathrm{ct}(F)Q_{m,p}/\mathrm{ct}(Q_{m,p}) \bmod p$ pour tout $p \in E$ et $\|Q_m\|_\infty \leq \sqrt{n+1}\, 2^n \|F\|_\infty$.
6. Remplacer les Q_m par leurs parties primitives et renvoyer la liste des paires (Q_m, m).

Algorithme 21.1 – Factorisation sans carré dans $\mathbb{Q}[X]$.

D'autre part le Théorème 21.6 de Landau–Mignotte permet de borner la taille des coefficients de tout facteur Q primitif de F par la formule suivante :

$$\|\mathrm{ct}(F)Q/\mathrm{ct}(Q)\|_\infty \leq \sqrt{n+1}\, 2^n \|F\|_\infty.$$

En prenant pour k le premier entier tel que

$$(3/2)^k > 2 \cdot 8^n \sqrt{n+1}\, n^{2n} \|F\|_\infty^{2n+1},$$

alors on a nécessairement

$$\prod_{\substack{n+1 \leq p \leq k \\ p \text{ premier}}} p > 2|\mathrm{ct}(F)M| \|\mathrm{ct}(F)Q/\mathrm{ct}(Q)\|_\infty,$$

en utilisant les bornes pour $|\mathrm{ct}(F)M|$ et $\|\mathrm{ct}(F)Q/\mathrm{ct}(Q)\|_\infty$ que nous venons de voir, ainsi que le Lemme 21.13. Par conséquent, parmi tous les calculs de fsc(F mod p) pour tout $n+1 \leq p \leq k$, on rencontre au moins un premier p compatible. Cela justifie que l'algorithme détermine bien le degré de la partie sans carré de F, et sélectionne ainsi tous les nombres premiers p compatibles. De plus, le choix de k garantit aussi que le produit des nombres premiers de E est deux fois supérieur à tous les coefficients des $\mathrm{ct}(F)Q/\mathrm{ct}(Q)$, pour tout facteur sans carré Q de F.

En notant $\eta = \|\mathrm{ct}(F)Q/\mathrm{ct}(Q)\|_\infty$ et $\pi = \prod_{p \in E} p$, on a une injection naturelle de $\{-\eta, \ldots, \eta\}$ dans $\mathbb{Z}/\pi\mathbb{Z}$. Un coefficient c de $\mathrm{ct}(F)Q/\mathrm{ct}(Q)$ est envoyé dans $\{0, \ldots, \eta\}$ s'il est

positif ou nul, et dans $\{\pi-\eta,\dots,\pi-1\}$ sinon. La condition $\pi > 2\eta$ assure que ces deux ensembles sont disjoints. L'utilisation des restes chinois à l'étape (5) permet donc bien de reconstruire les polynômes $\mathrm{ct}(F)Q/\mathrm{ct}(Q)$. L'algorithme est correct.

Le coût binaire des étapes (1) et (2) est $\tilde{O}(n\log\|F\|_\infty)$, par le Lemme 21.12. Les calculs multimodulaires et des restes chinois dans les étapes (4) et (5) ont coût $\tilde{O}(n^2\log\|F\|_\infty)$, par la Proposition 5.3. Pour les calculs des factorisations sans carré, on utilise le Théorème 18.16, qui reflète dans le présent cas le seul coût de l'Algorithme 18.2 de Yun, puisque $p \geq n+1$. L'étape (3) totalise un coût binaire $\tilde{O}(n^2\log\|F\|_\infty)$. Les parties primitives de la dernière étape s'obtiennent en coût $\tilde{O}(n(n+\log\|F\|_\infty))$. ■

21.3 Algorithme par remontée et recombinaison

Sans perte de généralité, à parir de maintenant, nous pouvons supposer F sans carré. Si p est un nombre premier qui ne divise pas $\mathrm{Res}(F,F')$, alors nous allons montrer qu'il existe une unique factorisation irréductible de F dans $\mathbb{Z}/p^\sigma\mathbb{Z}[X]$ pour chaque entier $\sigma \geq 1$. Lorsque $\sigma = 1$, il s'agit de la factorisation de F mod p, c'est-à-dire dans $\mathbb{F}_p[X]$, qui peut être calculée avec les algorithmes du Chapitre 19. Dans un premier temps nous allons montrer comment étendre cette factorisation pour n'importe quelle valeur de σ. Nous verrons que les facteurs de F dans $\mathbb{Z}/p^{\sigma+1}\mathbb{Z}[X]$ prolongent ceux dans $\mathbb{Z}/p^\sigma\mathbb{Z}[X]$. Nous parlerons alors de *remontée* des facteurs de F modulo p.

Ensuite, si $G \in \mathbb{Z}[X]$ divise F, alors les facteurs irreductibles de G dans $\mathbb{Z}/p^\sigma\mathbb{Z}[X]$ forment un sous-ensemble de ceux de F dans $\mathbb{Z}/p^\sigma\mathbb{Z}[X]$. Cette remarque nous mène alors à tester les produits de tous les sous-ensembles de facteurs de F dans $\mathbb{Z}/p^\sigma\mathbb{Z}[X]$, lorsque σ est suffisamment grand. C'est l'étape dite de *recombinaison*.

Choix de p

Pour choisir un premier p qui ne divise pas $\mathrm{Res}(F,F')$, il est plus efficace d'utiliser le lemme suivant plutôt que de calculer $\mathrm{Res}(F,F')$ puis de le réduire modulo p. Pour des raisons de complexité, nous sommes intéressés à avoir le plus petit nombre premier p qui ne divise pas $\mathrm{Res}(F,F')$.

Proposition 21.17 La recherche du plus petit nombre premier p qui ne divise pas $\mathrm{Res}(F,F')$ peut s'effectuer en $\tilde{O}(n^2\log\|F\|_\infty)$ opérations binaires. De plus, ce nombre premier est borné par $\tilde{O}(n\log\|F\|_\infty)$.

Démonstration. Par le Corollaire 21.11, la valeur absolue de $\mathrm{Res}(F,F')$ est au plus $\|F\|_2^{n-1}\|F'\|_2^n$. En prenant pour k le premier entier plus grand que

$$\log(\|F\|_2^{n-1}\|F'\|_2^n) = \tilde{O}(n\log\|F\|_\infty),$$

on calcule alors tous les nombres premiers inférieurs à k et on cherche le plus petit d'entre eux qui ne divise pas $\mathrm{Res}(F,F')$. Si aucun nombre premier ne convient, alors on double k et on cherche parmi les nouveaux nombres premiers. Le Lemme 21.13 nous assure que l'algorithme s'arrête avec k borné par $\tilde{O}(n\log\|F\|_\infty)$. Le Lemme 21.12 nous garantit que le calcul des nombres premiers inférieurs à k est quasi-linéaire en k.

Avec l'algorithme rapide utilisant l'arbre des sous-produits du Chapitre 5 (voir Théorème 5.3), on peut calculer F mod p pour tout nombre premier $p \leq k$ en temps

quasi-linéaire. Ensuite pour chaque p qui ne divise pas $\mathrm{ct}(F)$ on calcule le discriminant de F mod p en temps $\tilde{O}(n \log p)$. Au total on obtient un coût binaire $\tilde{O}(n^2 \log \|F\|_\infty)$. ■

Remontée de Newton–Hensel

Dans cette sous-section, $\mathbb{A}$ représente un anneau commutatif avec une unité, I est un idéal propre de $\mathbb{A}$, et F est un polynôme de $\mathbb{A}[X]$ de coefficient de tête noté $\mathrm{ct}(F)$, qui est supposé inversible modulo I. Le lemme suivant est une forme du lemme de Hensel, mais avec le point de vue calculatoire de l'itération de Newton.

Lemme 21.18 Supposons $\mathrm{Res}(F, F')$ inversible modulo I. Si $\mathfrak{g} \in \mathbb{A}[X]$ divise F modulo I et que $\mathrm{ct}(\mathfrak{g}) = 1$, alors il existe un unique polynôme $\mathfrak{G}$ congru à $\mathfrak{g}$ modulo I, unitaire et divisant F modulo I^2.

Démonstration. Posons $\mathfrak{h} = F/\mathfrak{g}$. Nous cherchons deux polynômes $\mathfrak{G}$ et $\mathfrak{H}$ congrus respectivement à $\mathfrak{g}$ et $\mathfrak{h}$ modulo I, avec $\mathfrak{G}$ unitaire, et tels que $F = \mathfrak{G}\mathfrak{H} \bmod I^2$. Cette dernière équation se récrit en $F = \mathfrak{g}\mathfrak{h} + \mathfrak{g}(\mathfrak{H} - \mathfrak{h}) + \mathfrak{h}(\mathfrak{G} - \mathfrak{g}) \bmod I^2$. En prenant les restes de la division par $\mathfrak{g}$ nous obtenons $\mathrm{rem}(F, \mathfrak{g}) = \mathrm{rem}(\mathfrak{h}(\mathfrak{G} - \mathfrak{g}), \mathfrak{g}) \bmod I^2$. Puisque le discriminant de F est inversible modulo I, le résultant de $\mathfrak{g}$ et $\mathfrak{h}$ est inversible modulo I, et donc $\mathfrak{h}$ est inversible modulo $\mathfrak{g}$ dans $(\mathbb{A}/I^2)[X]$. En notant $\mathfrak{u}$ son inverse, nous en déduisons la formule suivante :

$$\mathfrak{G} = \mathfrak{g} + \mathrm{rem}(\mathfrak{u}F, \mathfrak{g}) \bmod I^2. \qquad ■$$

Comme $F' = \mathfrak{g}'\mathfrak{h} + \mathfrak{g}\mathfrak{h}' \bmod I$, alors $\mathfrak{h} = F'/\mathfrak{g}' \bmod \mathfrak{g} \bmod I$ et nous obtenons la formule

$$\mathfrak{G} - \mathfrak{g} = \frac{\mathfrak{g}'}{F'} F \bmod \mathfrak{g} \bmod I^2,$$

et par conséquent un algorithme pour calculer $\mathfrak{G}$. Une partie importante du calcul consiste à évaluer F et F′ modulo $\mathfrak{g}$ et I^2 comme dans le cas de l'opérateur de Newton.

En fait, comme F mod $\mathfrak{g}$ a tous ses coefficients dans I, il est suffisant de connaître l'inverse de F′ modulo $\mathfrak{g}$ à la précision I. Lorsque I est maximal, $\mathbb{A}/I$ est un corps et nous pouvons obtenir cet inverse à partir de l'algorithme du pgcd étendu. En général, l'inverse $\mathfrak{U}$ de F′ modulo $\mathfrak{G}$ à la précision I^2 peut être déduit de l'inverse $\mathfrak{u}$ de F′ à la précision I par la formule suivante :

$$\mathfrak{U} = \mathfrak{u} - \mathfrak{u}(\mathfrak{u}F' - 1) \bmod \mathfrak{G},$$

puisque $\mathfrak{U}F' = 1 - (\mathfrak{u}F' - 1)^2 = 1 \bmod \mathfrak{G} \bmod I^2$. Cette méthode est résumée dans l'Algorithme 21.2.

Proposition 21.19 Soient $\mathbb{A} = \mathbb{Z}$ et $I = (p)$ avec p premier. À partir d'une décomposition $F = \mathrm{ct}(F)\mathfrak{f}_1 \cdots \mathfrak{f}_s$ modulo p, avec des $\mathfrak{f}_i$ unitaires, nous pouvons calculer la decomposition $F = \mathrm{ct}(F)\mathfrak{F}_1 \cdots \mathfrak{F}_s$ modulo p^σ telle que $\mathfrak{F}_i = \mathfrak{f}_i \bmod p$ pour tout $i \in \{1, \ldots, m\}$, en temps $O(M_{\mathbb{Z}}(\sigma \log p) M(d) \log d)$.

Démonstration. Il suffit d'appliquer successivement l'Algorithme 21.2 pour arriver en précision p^2, p^4,... jusqu'à dépasser p^σ. Le coût découle principalement de la

Entrée $F \in \mathbb{A}[X]$, et $\mathfrak{f}_1, \ldots, \mathfrak{f}_s$ unitaires dans $(\mathbb{A}/I)[X]$ tels que $F = \mathrm{ct}(F)\mathfrak{f}_1 \cdots \mathfrak{f}_s$ dans $(\mathbb{A}/I)[X]$.
Sortie $\mathfrak{F}_1, \ldots, \mathfrak{F}_s$ unitaires dans $(\mathbb{A}/I^2)[X]$ tels que $F = \mathrm{ct}(F)\mathfrak{F}_1 \cdots \mathfrak{F}_s$ dans $(\mathbb{A}/I^2)[X]$ et $\mathfrak{F}_i = \mathfrak{f}_i \bmod I$ pour tout i.

1. Calculer $F \bmod \mathfrak{f}_1, \ldots, F \bmod \mathfrak{f}_s$ et $F' \bmod \mathfrak{f}_1, \ldots, F' \bmod \mathfrak{f}_s$ modulo I^2.
2. Calculer $\mathfrak{r}_1 = \mathfrak{f}_1' \frac{F}{F'} \bmod \mathfrak{f}_1, \ldots, \mathfrak{r}_s = \mathfrak{f}_s' \frac{F}{F'} \bmod \mathfrak{f}_s$ modulo I^2.
3. Renvoyer $\mathfrak{f}_1 + r_1, \ldots, \mathfrak{f}_s + r_s$.

Algorithme 21.2 – Remontée de Hensel simultanée.

Proposition 5.6, auquel il faut ajouter le coût pour obtenir les inverses de $F' \bmod \mathfrak{f}_i$ pour tout i, c'est-à-dire $O(\mathsf{M}_{\mathbb{Z}}(\sigma \log p)\mathsf{M}(d)\log d)$. ■

Proposition 21.20 Soient $\mathbb{K}$ un corps, $\mathbb{A} = \mathbb{K}[t]$ et $I = (t)$. À partir d'une décomposition $F = \mathrm{ct}(F)\mathfrak{f}_1 \cdots \mathfrak{f}_s$ modulo t, avec des $\mathfrak{f}_i$ unitaires, nous pouvons calculer la decomposition $F = \mathrm{ct}(F)\mathfrak{F}_1 \cdots \mathfrak{F}_s$ modulo t^σ telle que $\mathfrak{F}_i = \mathfrak{f}_i \bmod t$ pour tout $i \in \{1, \ldots, m\}$, en utilisant $O(\mathsf{M}(\sigma)\mathsf{M}(d)\log d)$ opérations arithmétiques dans $\mathbb{K}$.

Démonstration. La preuve est similaire à la précédente. ■

Chaque élément a de $\mathbb{Z}/p^\sigma\mathbb{Z}$ s'écrit de façon unique en base p sous la forme $\sum_{i=0}^{\sigma-1} a_i p^i$, avec les a_i dans $\{0, \ldots, p-1\}$. En s'inspirant de la construction des anneaux des séries formelles, on peut faire tendre σ vers l'infini et définir l'anneau des *entiers p-adiques*, noté $\mathbb{Z}_p$, comme l'ensemble des limites des tels développements en base p. Les éléments sont de la forme $\sum_{i=0}^{+\infty} a_i p^i$. Les factorisations de F dans $\mathbb{Z}/p^\sigma\mathbb{Z}$ convergent donc vers l'unique factorisation irréductible p-adique de F dans $\mathbb{Z}_p[X]$.

Exemple 21.2 Avec $F(X) = X^4 - 1$, on peut prendre $p = 5$. Les facteurs irréductibles dans $\mathbb{Z}/p\mathbb{Z}[X]$ sont $X+1$, $X+2$, $X+3$ et $X+4$. Avec $\sigma = 10$, on a $\mathfrak{F}_1(X) = X + 1 + O(p^{10})$, $\mathfrak{F}_2(X) = X + 2 + p + 2p^2 + p^3 + 3p^4 + 4p^5 + 2p^6 + 3p^7 + 3p^9 + O(p^{10})$, $\mathfrak{F}_3(X) = X + 3 + 3p + 2p^2 + 3p^3 + p^4 + 2p^6 + p^7 + 4p^8 + p^9 + O(p^{10})$, $\mathfrak{F}_4(X) = X + 4 + 4p + 4p^2 + 4p^3 + 4p^4 + 4p^5 + 4p^6 + 4p^7 + 4p^8 + 4p^9 + O(p^{10})$.

Recombinaison des facteurs p-adiques

Notons $\mathfrak{F}_1, \ldots, \mathfrak{F}_s$ les facteurs irréductibles unitaires de F dans $\mathbb{Z}/p^\sigma\mathbb{Z}$, de sorte que $F = \mathrm{ct}(F)\mathfrak{F}_1 \cdots \mathfrak{F}_s$. Si $G \in \mathbb{Z}[X]$ est un diviseur propre et primitif de F, alors il existe un sous-ensemble E de $\{1, \ldots, s\}$ tels que $G(X) = \mathrm{ct}(G)\prod_{e \in E} \mathfrak{F}_e$. Comme il n'est pas souhaitable de factoriser ct(F) dans $\mathbb{Z}$ de sorte à énumérer toutes les valeurs possibles pour ct(G), on va examiner $H(X) = \mathrm{ct}(F)G(X)/\mathrm{ct}(G) = \mathrm{ct}(F)\prod_{e \in E} \mathfrak{F}_e$. Il s'agit d'un polynôme de $\mathbb{Z}[X]$ dont les normes des coefficients sont au plus $\eta = \sqrt{n+1}\, 2^m \|F\|_\infty$ par le Théorème 21.6.

Entrée $F \in \mathbb{Z}[X]$, séparable et primitif.
Sortie Les facteurs irréductibles de F.

1. Trouver le plus petit nombre premier p qui ne divise pas $\mathrm{Res}(F, F')$.
2. Déterminer le premier entier σ tel que $p^\sigma > 2\eta$, avec $\eta = \sqrt{n+1}\, 2^n \|F\|_\infty$.
3. Calculer les facteurs unitaires irréductibles $f_1, \ldots, f_s$ de F dans $\mathbb{Z}/p\mathbb{Z}[X]$.
4. Remonter la factorisation irréductible $f_1, \ldots, f_s$ de F dans $\mathbb{Z}/p^\sigma\mathbb{Z}[X]$ à l'aide de l'Algorithme 21.2.
5. Initialiser S avec $\{1, \ldots, s\}$, puis T avec $\{\}$, et L avec la liste vide.
6. Parcourir les sous-ensembles E non vides de S par cardinal croissant au plus $(s - \mathrm{card}(T))/2$ et disjoints de T :
   a. calculer $\mathrm{ct}(F)\prod_{e\in E} f_e$ modulo p^σ, puis sa préimage H dans $\mathbb{Z}[X]$;
   b. calculer la partie primitive G de H ;
   c. si G divise F alors l'insérer dans L, remplacer T par $T \cup E$, et F par F/G.
7. Si $\deg(F) \geq 1$ alors insérer F dans L.
8. Renvoyer L.

Algorithme 21.3 – Méthode naïve de factorisation par remontée et recombinaison.

On fixe maintenant σ au premier entier tel que

$$p^\sigma > 2\eta = \sqrt{n+1}\, 2^{m+1} \|F\|_\infty.$$

De cette manière, on a une injection

$$\{-\eta, \ldots, \eta\} \hookrightarrow \mathbb{Z}/p^\sigma\mathbb{Z}.$$

Un coefficient h_i de H est envoyé dans $\{0, \ldots, \eta\}$ s'il est positif ou nul, et dans $\{-\eta + p^\sigma, \ldots, p^\sigma - 1\}$ sinon. La condition $p^\sigma > 2\eta$ assure que ces deux ensembles sont disjoints. Par conséquent à partir de $\mathrm{ct}(F)\prod_{e\in E} f_e$ calculé modulo p^σ on peut reconstruire le polynôme $H(X) = \mathrm{ct}(F)G(X)/\mathrm{ct}(G)$, puis déduire G comme la partie primitive de H. Cette approche conduit à l'Algorithme 21.3.

Proposition 21.21 L'Algorithme 21.3 est correct et a pour coût binaire

$$\tilde{O}(2^n \log \|F\|_\infty).$$

Démonstration. Avec $p = \tilde{O}(n \log \|F\|_\infty)$ tel que spécifié dans la Proposition 21.17, la Proposition 19.13 permet de réaliser la factorisation de l'étape (3) de façon déterministe en $\tilde{O}(n^3 + pn^2) = \tilde{O}(n^3 \log \|F\|_\infty)$ opérations binaires. Ensuite, le coût de l'étape (4) est quasi-linéaire, grâce à la Proposition 21.19.

Pour le calcul de H, on construit l'arbre des sous-produits présenté dans la Section 5.4 de sorte que la Proposition 5.5 implique un coût binaire quasi-linéaire. Pour le calcul du contenu de H on peut utiliser successivement n fois l'algorithme rapide de pgcd, pour obtenir aussi un coût quasi-linéaire. Pour tester si G divise F on peut calculer F/G dans $\mathbb{Z}/p^\sigma\mathbb{Z}[X]$ puis la préimage Q de ce quotient dans $\mathbb{Z}[X]$ et enfin vérifier que F est égal à GQ. De cette manière le coût est quasi-linéaire.

Enfin, le nombre de candidats G à construire et tester dans l'étape (6), tant que F et S ne changent pas, est borné par le nombre de sous-ensembles de $S \setminus T$ de cardinal au plus $(s - \mathrm{card}(T))/2$, c'est-à-dire $2^{s-\mathrm{card}(T)-1}$. ■

Prendre pour p le plus petit nombre premier qui ne divise pas $\mathrm{Res}(F, F')$ est une bonne stratégie pour l'analyse de la complexité, mais ce n'est pas le meilleur choix en pratique, où il vaut mieux que p soit de taille proche de celle d'un registre, c'est-à-dire 32 ou 64 bits avec les processeurs actuels.

Exemple 21.3 Avec $F(X) = X^4 - 1$, et $p = 5$, comme dans l'Exemple 21.2, on a $\eta = 16\sqrt{5} < 36$, et il est suffisant de prendre $\sigma = 3$. Avec $E = \{1\}$, on obtient que $G(X) = H(X) = X + 1$ est un facteur irréductible de F. Ensuite $E = \{2\}$ donne $G(X) = H(X) = X + 57$ qui ne divise pas F. Avec $E = \{3\}$, on a $G(X) = H(X) = X - 59$ qui ne divise pas F. Avec $E = \{4\}$, on a $G(X) = H(X) = X - 1$ qui divise F. Le dernier facteur irréductible de F est donc $X^2 + 1$, obtenu avec $E = \{2, 3\}$.

R L'Algorithme 21.3 peut être amélioré. Déjà, lorsque F et T changent à l'étape (6), il est inutile de reconsidérer les ensembles E déjà traités. Ensuite, à chaque tel changement on peut recalculer une nouvelle précision σ, plus petite. Aussi on peut se contenter de chercher les facteurs irréductibles G de degrés au plus $d/2$. S'il n'existe pas de tels facteurs alors F est irréductible. Enfin le test de divisibilité de F par G doit être optimisé de sorte à détecter au plus vite la non-divisibilité, qui est le cas le plus fréquent rencontré au cours de l'algorithme. Par exemple on peut montrer que G ne divise pas F modulo un nombre premier $p' \neq p$.

En pratique il est aussi intéressant d'essayer plusieurs valeurs de p à l'étape (1), de sorte à minimiser le nombre s de facteurs p-adiques. En général on observe souvent que s est plutôt petit par rapport à n. Du coup cet algorithme est raisonnablement efficace, tant que s ne dépasse pas la vingtaine. Néanmoins il existe certains cas extrêmes où $n = O(s)$, comme dans l'exemple suivant.

Exemple 21.4 Soit p_i le i-ième nombre premier pour $i \geq 1$. Le m-ième polynôme de *Swinnerton–Dyer* est défini par

$$S_m(X) = \prod \left(X \pm \sqrt{2} \pm \sqrt{3} \pm \sqrt{5} \pm \cdots \pm \sqrt{p_m}\right),$$

où le produit est pris sur les 2^m possibilités de signes + et −. Le degré de S_m est $n = 2^m$. Ces polynômes peuvent être calculés numériquement (par exemple avec une arithmétique d'intervalle pour garantir le calcul) ou bien avec une cascade de résultants, comme par exemple : $S_2(X) = \mathrm{Res}_{z_2}(\mathrm{Res}_{z_1}(X - z_1 - z_2, z_1^2 - 2), z_2^2 - 3)$. Par cette dernière formule nous constatons que S_m a tous ses coefficients dans $\mathbb{Z}$ et qu'il est unitaire. Par des arguments classiques de théorie de Galois, nous

savons que S_m est irréductible dans $\mathbb{Z}[X]$. Par ailleurs, si p est un nombre premier alors $\mathbb{F}_{p^2}$ contient toutes les racines carrées modulo p. Par conséquent $S_m(X)$ se factorise en facteurs de degrés 1 ou 2 modulo p. Le temps pour retrouver les recombinaisons des facteurs p-adiques est donc exponentiel en n. Avec $m = 3$, on a $S_3(X) = X^8 - 40X^6 + 352X^4 - 960X^2 + 576$. La factorisation modulo $p = 2$ est X^8, modulo $p = 3$ on a $(X^2+1)^2X^4$, modulo $p = 5$ on a $(X^2+1)^2X^4$, modulo $p = 7$ on a $(X^2+6X+3)(X^2+X+6)(X^2+6X+6)(X^2+X+3)$, etc.

Dans certaines situations, comme celles rencontrées au Chapitre 16, on souhaite connaître les racines entières ou rationnelles d'un polynôme F de $\mathbb{Z}[X]$. Il n'est alors pas utile de calculer toute la factorisation irréductible de F mais seulement ses facteurs de degré 1. À cet effet l'Algorithme 21.3 peut être naturellement simplifié pour conduire à la complexité suivante :

Proposition 21.22 Le calcul des racines dans $\mathbb{Q}$ d'un polynôme F de $\mathbb{Z}[X]$, primitif et séparable, de degré n peut se faire en coût binaire $\tilde{O}(n^2 \log \|F\|_\infty)$.

Démonstration. Il s'agit tout simplement de restreindre l'Algorithme 21.3 à ne chercher que les facteurs de degré 1, ce qui correspond à ne considérer que des sous-ensembles E de cardinal 1 dans l'étape (6). Par conséquent, dans l'étape (3) on peut se contenter de ne calculer que les facteurs linéaires de F modulo p. Cela peut être fait en évaluant F sur tous les points de $\mathbb{Z}/p\mathbb{Z}$. Comme $p = \tilde{O}(n \log \|F\|_\infty)$, l'algorithme d'évaluation multipoint du Théorème 5.1 permet d'effectuer cette tâche au moyen de $\tilde{O}(n \log \|F\|_\infty)$ opérations dans $\mathbb{Z}/p\mathbb{Z}$, ce qui donne un coût binaire $\tilde{O}(n \log \|F\|_\infty)$. La complexité découle de l'analyse faite dans la preuve de la proposition précédente. ■

21.4 Algorithme de van Hoeij

Nous présentons dans cette section un algorithme de factorisation déterministe fonctionnant en temps polynomial. Comme pour l'algorithme précédent, nous commençons par calculer une factorisation p-adique, mais jusqu'à une précision plus élevée de sorte à pouvoir résoudre le problème de recombinaison en temps polynomial, au moyen de l'algorithme LLL du chapitre précédent. L'idée clé est la suivante : si F se factorise en $F_1 \cdots F_r$ alors l'égalité

$$F' = F\frac{F_1'}{F_1} + \cdots + F\frac{F_r'}{F_r}$$

permet, d'une certaine façon, de « linéariser » le problème.

Dérivées logarithmiques

Continuons à supposer que p est un nombre premier qui ne divise pas $\mathrm{Res}(F, F')$, et notons $f_1, \ldots, f_s$ les facteurs p-adiques unitaires de F en précision σ. Pour chaque facteur irréductible F_i de F on définit le vecteur $\mu_i \in \{0,1\}^s$ par l'égalité

$$F_i = \mathrm{ct}(F_i) f_1^{\mu_{i,1}} \cdots f_s^{\mu_{i,s}} + O(p^\sigma).$$

En prenant la dérivée logarithmique et en multipliant par F les deux membres de cette égalité, nous obtenons :

$$F\frac{F_i'}{F_i} = \mu_{i,1}F\frac{f_1'}{f_1} + \cdots + \mu_{i,s}F\frac{f_s'}{f_s} + O(p^\sigma).$$

On doit donc chercher des entiers $\ell_1, \ldots, \ell_s$ aussi petits que possible tels que

$$\ell_1 F\frac{f_1'}{f_1} + \cdots + \ell_s F\frac{f_s'}{f_s}$$

soit un polynôme à coefficients des entiers pas trop grands modulo p^σ. Dans un premier temps il nous faut donc borner la taille des coefficients de tels polynômes lorsqu'ils sont de la forme FG′/G, avec G un diviseur de F dans $\mathbb{Z}[X]$. En utilisant successivement la Proposition 21.5, la multiplicativité de la mesure de Mahler M, le Théorème 21.7, puis la Proposition 21.4, on obtient :

$$\begin{aligned}\|FG'/G\|_2 &\le 2^{n-1}M(FG'/G) = 2^{n-1}M(F/G)M(G') \\ &\le n2^{n-1}M(F/G)M(G) = n2^{n-1}M(F) \le n2^{n-1}\|F\|_2.\end{aligned}$$

À partir de maintenant on prend pour τ le premier entier tel que $p^\tau > n2^n\|F\|_2$. Ainsi, pour tout $i \in \{1, \ldots, r\}$, on peut reconstruire $F\frac{F_i'}{F_i}$ à partir de sa projection dans $\mathbb{Z}/p^\tau\mathbb{Z}[X]$.

Troncatures signées

Si $\mathfrak{a} = \sum_{i\ge 0} a_i p^i$ est un nombre p-adique, nous notons $\lceil \mathfrak{a} \rceil^\sigma$ l'entier $\sum_{i=0}^{\sigma-1} a_i p^i$. On définit aussi le *reste signé* dans la division par p^τ l'entier $r = \sum_{i=0}^{\tau-1} a_i p^i$ si $r \le p^\tau/2$, et $r - p^\tau$ sinon. On définit alors $\lfloor \mathfrak{a} \rfloor_\tau$ comme l'entier p-adique $(\mathfrak{a} - r)/p^\tau$, de sorte que

$$\mathfrak{a} = \lfloor \mathfrak{a} \rfloor_\tau p^\tau + r \quad \text{avec} \quad r \in \left]-\frac{p^\tau}{2}, \frac{p^\tau}{2}\right].$$

Enfin on utilisera la notation $[\mathfrak{a}]_\tau^\sigma = \lceil \lfloor \mathfrak{a} \rfloor_\tau \rceil^\sigma$, qui satisfait $\mathfrak{a} = [\mathfrak{a}]_\tau^\sigma p^\tau + r + O(p^\sigma)$.

Lorsqu'il est positif, un nombre entier c peut simplement être vu comme un entier p-adique en considérant son écriture en base p. Lorsqu'il est négatif, son image dans $\mathbb{Z}_p$ peut être obtenue comme l'opposé de l'image de $-c$. Par exemple avec $p = 5$, l'image de -7 est $3 + 3p + 4p^2 + 4p^3 + 4p^4 + 4p^5 + \cdots$. Dans la suite, l'écriture $[c]_\tau^\sigma$ représentera $[\mathfrak{a}]_\tau^\sigma$ en prenant pour $\mathfrak{a}$ l'image de c dans $\mathbb{Z}_p$. Aussi, si $\mathcal{Q} = \sum_{i=0}^d \mathcal{Q}_i X^i \in \mathbb{Z}_p[X]$, alors on notera, par extension, $[\mathcal{Q}]_\tau^\sigma = \sum_{i=0}^d [\mathcal{Q}_i]_\tau^\sigma X^i \in \mathbb{Z}[X]$.

La fonction $[\cdot]_\tau^\sigma$ n'est pas additive à cause des retenues. Si $\mathfrak{a}_1, \ldots, \mathfrak{a}_n$ sont des entiers p-adiques, et si $\ell_1, \ldots, \ell_n$ sont des entiers positifs ou négatifs, alors on peut écrire

$$[\ell_1\mathfrak{a}_1 + \cdots + \ell_n\mathfrak{a}_n]_\tau^\sigma = [\ell_1\mathfrak{a}_1]_\tau^\sigma + \cdots + [\ell_n\mathfrak{a}_n]_\tau^\sigma + \epsilon + O(p^\sigma),$$

avec $\epsilon \in \mathbb{Z}$ borné par $|\epsilon| \le (\ell_1 + \cdots + \ell_n)/2$.

Construction du réseau

Posons $m = s + n$ et considérons le réseau L de $\mathbb{Z}^m$ engendré par les lignes de la matrice

$$\begin{pmatrix} 1 & & [\mathfrak{G}_{1,0}]_\tau^\sigma & \cdots & [\mathfrak{G}_{1,d-1}]_\tau^\sigma \\ & \ddots & \vdots & & \vdots \\ & & 1 & [\mathfrak{G}_{s,0}]_\tau^\sigma & \cdots & [\mathfrak{G}_{s,d-1}]_\tau^\sigma \\ & & & p^{\sigma-\tau} & & \\ & & & & \ddots & \\ & & & & & p^{\sigma-\tau} \end{pmatrix}, \tag{21.1}$$

où $\mathfrak{G}_{i,j}$ représente le coefficient de degré j de $\mathrm{F}\frac{\mathsf{F}_i'}{\mathsf{F}_i}$. Pour expliquer en quoi cette construction est pertinente, commençons par montrer que L contient des vecteurs « courts » provenant des facteurs irréductibles de F. Par le choix de τ on sait que

$$\left\lfloor \mu_{i,1}\mathrm{F}\frac{\mathsf{F}_1'}{\mathsf{F}_1} + \cdots + \mu_{i,s}\mathrm{F}\frac{\mathsf{F}_s'}{\mathsf{F}_s} \right\rfloor_\tau = \mathrm{O}(p^\sigma),$$

et donc pour toute valeur de $\sigma \geq \tau$ on obtient

$$\left[\mu_{i,1}\mathrm{F}\frac{\mathsf{F}_1'}{\mathsf{F}_1} + \cdots + \mu_{i,s}\mathrm{F}\frac{\mathsf{F}_s'}{\mathsf{F}_s} \right]_\tau^\sigma = \mathrm{O}(p^{\sigma-\tau}),$$

ce qui fournit un polynôme $\epsilon_i \in \mathbb{Z}[\mathrm{X}]$ de norme $\|\epsilon_i\|_\infty \leq s/2$, tel que

$$\mu_{i,1}\left[\mathrm{F}\frac{\mathsf{F}_1'}{\mathsf{F}_1}\right]_\tau^\sigma + \cdots + \mu_{i,s}\left[\mathrm{F}\frac{\mathsf{F}_s'}{\mathsf{F}_s}\right]_\tau^\sigma + \delta_i p^{\sigma-\tau} = \epsilon_i,$$

où $\delta_i \in \mathbb{Z}[\mathrm{X}]$ accumule les retenues à partir de la précision $\sigma - \tau$. Par conséquent on vient de montrer que les vecteurs $(\mu_{i,1}, \ldots, \mu_{i,s}, \epsilon_{i,0}, \ldots, \epsilon_{i,n-1})$, notés plus simplement (μ_i, ϵ_i), sont dans L. Ils peuvent être qualifiés de « courts » puisqu'ils ont pour norme

$$\|(\mu_i, \epsilon_i)\|_2 \leq \sqrt{s + n(s/2)^2}.$$

Posons $\eta = \sqrt{s + n(s/2)^2}$, et définissons W comme le réseau engendré par

$$(\mu_1, \epsilon_1), \ldots, (\mu_r, \epsilon_r),$$

de sorte que $\mathrm{W} \subseteq \mathrm{L}$. L'étape suivante de la méthode de factorisation consiste à calculer une base de W à partir de L.

Réduction du réseau

Nous notons $w_1, \ldots, w_m$ la base réduite de L obtenue en utilisant l'Algorithme 20.2 de réduction LLL du chapitre précédent. Nous souhaitons pouvoir en déduire les μ_i dès que σ est suffisamment grand. Nous introduisons ensuite

$$t = \min\{i \mid \|v_j^*\|_2 > \eta,\ \forall j > i\}.$$

Le lemme suivant permet de restreindre L à un sous-réseau $\tilde{\mathrm{L}}$ qui contient toujours W.

Lemme 21.23 Si v est un vecteur de L de norme $\|v\|_2 \leq \eta$, alors il appartient au réseau $\tilde{L}$ engendré par $w_1, \ldots, w_t$. En particulier on a $W \subseteq \tilde{L}$.

Démonstration. Notons $i \leq m$ le plus petit entier tel que v soit dans réseau engendré par $w_1, \ldots, w_i$. Par définition du procédé d'orthogonalisation de Gram–Schmidt, c'est aussi le premier entier tel que v est dans l'espace vectoriel engendré par $w_1^*, \ldots, w_i^*$, de sorte que

$$v = \ell_i w_i + \cdots + \ell_1 w_1 = \ell_i' w_i^* + \cdots + \ell_1' w_1^*,$$

avec $\ell_i' = \ell_i \geq 1$. Par conséquent $\|v\|_2 \geq \|w_i\|_2$, et nécessairement $i \leq t$. ■

Reconstruction des facteurs

Par définition de t, on a $\|w_t^*\|_2 \leq \eta$. Par ailleurs, la Proposition 20.3 nous donne $\|w_i\|_2^2 \leq 2^{t-1}\|w_t^*\|_2^2$ pour tout $i \leq t$. Par conséquent, pour tout $i \leq t$, on a

$$\|w_i\|_2 \leq 2^{(t-1)/2}\eta \leq 2^{(m-1)/2}\eta.$$

Fixons maintenant un indice $i \leq t$ et examinons w_i. Fixons ensuite un indice j de w_i tel que $w_{i,j} \neq 0$. Il existe un unique entier k tel que $\mu_{k,j} = 1$. On construit alors le vecteur

$$\tilde{w}_i = w_i - w_{i,j}(\mu_k, \epsilon_k),$$

qui satisfait $\tilde{w}_{i,j} = 0$ et a pour norme

$$\|\tilde{w}_i\|_2 \leq \|w_i\|_2 + \|w_i\|_2\eta \leq 2^{(m-1)/2}(1+\eta)\eta.$$

Comme $\tilde{w}_i$ est dans le réseau L, on a

$$\tilde{w}_{i,1}\left[F\frac{F_1'}{F_1}\right]_\tau^\sigma + \cdots + \tilde{w}_{i,s}\left[F\frac{F_s'}{F_s}\right]_\tau^\sigma = \phi_i + O(p^{\sigma-\tau}),$$

avec ϕ_i un polynôme de $\mathbb{Z}[X]$ de degré au plus $n-1$ et de norme

$$\|\phi_i\|_2 \leq \|\tilde{w}_i\|_2 \leq 2^{(m-1)/2}(1+\eta)\eta.$$

On considère ensuite le polynôme

$$\mathfrak{H}_i = \tilde{w}_{i,1}F\frac{F_1'}{F_1} + \cdots + \tilde{w}_{i,s}F\frac{F_s'}{F_s},$$

et on écrit

$$[\mathfrak{H}_i]_\tau^\sigma = \tilde{w}_{i,1}\left[F\frac{F_1'}{F_1}\right]_\tau^\sigma + \cdots + \tilde{w}_{i,s}\left[F\frac{F_s'}{F_s}\right]_\tau^\sigma + \psi_i + O(p^{\sigma-\tau}),$$

avec ψ_i un polynôme de $\mathbb{Z}[X]$ de degré au plus $n-1$ et de norme

$$\|\psi_i\|_\infty \leq (|\tilde{w}_{i,1}| + \cdots + |\tilde{w}_{i,s}|)/2 \leq \sqrt{n}\,\|\tilde{w}_i\|_2/2 \leq \sqrt{n}\,2^{(m-3)/2}(1+\eta)\eta.$$

On écrit ensuite

$$\mathfrak{H}_i = [\mathfrak{H}_i]_\tau^\sigma p^\tau + \rho_i + O(p^\sigma) = (\phi_i + \psi_i)p^\tau + \rho_i + O(p^\sigma)$$

avec $\|\rho_i\|_\infty \leq p^\tau/2$. On pose $H_i = (\phi_i + \psi_i)p^\tau + \rho_i$. Il s'agit d'un polynôme de $\mathbb{Z}[X]$ de norme

$$\|H_i\|_\infty \leq \left(\left(1 + \frac{\sqrt{n}}{2}\right)2^{(m-1)/2}(1+\eta)\eta + \frac{1}{2}\right)p^\tau,$$

et tel que $H_i = \mathfrak{H}_i + O(p^\sigma)$. On pose

$$\theta = \sqrt{n}\left(\left(1 + \frac{\sqrt{n}}{2}\right)2^{(m-1)/2}(1+\eta)\eta + \frac{1}{2}\right)p^\tau$$

de manière à avoir $\|H_i\|_2 \leq \theta$.

Comme $\mathfrak{F}_j$ divise $\mathfrak{H}_i$ et F à la précision p^σ, alors le résultant $R_i = \mathrm{Res}(H_i, F)$ vaut $O(p^\sigma)$. On utilise par ailleurs le Corollaire 21.11

$$|R_i| \leq \|H_i\|_2^n \|F\|_2^{n-1} \leq \theta^n \|F\|_2^{n-1}.$$

La condition $p^\sigma > 2|R_i|$ garantit alors que $R_i = 0$. Par conséquent G_k divise H_i, ce qui implique que $w_{i,l} = w_{i,j}$ pour tout indice l tel que $\mu_{k,l} = 1$. En faisant de même avec les autres valeurs de j on déduit que, pour tout $i \leq t$, $\tilde{w}_i$ est dans l'espace engendré par $\mu_1, \ldots, \mu_r$. On vient donc de prouver que $L' = W$.

Pour retrouver les μ_i à partir de $w_1, \ldots, w_t$, on peut peut calculer la forme échelonnée sur $\mathbb{Q}$ de la matrice obtenue en mettant les w_i en lignes les uns au dessus des autres. Nous renvoyons le lecteur au Chapitre 8 pour les détails.

Synthèse de l'algorithme

Le méthode de factorisation est résumée dans l'Algorithme 21.4.

Théorème 21.24 L'Algorithme 21.4 de factorisation est correct et requiert $\tilde{O}(n^7(n + \log\|F\|_\infty)^2)$ opérations binaires.

Démonstration. Nous venons de voir tout au long de cette section que l'algorithme est correct. Il nous reste à analyser sa complexité. Le coût des étapes de (1) à (5) s'analyse comme pour l'Algorithme 21.3 : il est borné par $O(n^3 \log\|F\|_\infty)$, car $\tau \log p = \tilde{O}(n + \log\|F\|_\infty)$, et $\sigma \log p = \tilde{O}(n^2 + n\log\|F\|_\infty)$.

La construction de la matrice de l'étape (6) requiert $\tilde{O}(n^2 \sigma \log p)$ opérations, ce qui est dominé par la réduction du réseau par LLL qui coûte ensuite $\tilde{O}(n^5(\sigma \log p)^2) = \tilde{O}(n^5(n^2 + n\log\|F\|_\infty)^2)$. Le calcul de la forme échelonnée est négligeable dans ce cas particulier. Enfin la reconstruction des facteurs peut se faire en temps quasi-linéaire. ■

R Lorsque s est petit il est intéressant d'utiliser l'Algorithme 21.3 naïf de recombinaison. Aussi, avant d'utiliser l'algorithme de van Hoeij, il est utile de tester si

Entrée $F \in \mathbb{Z}[X]$, séparable et primitif.
Sortie Les facteurs irréductibles de F.

1. Trouver le plus petit nombre premier p qui ne divise pas $\mathrm{Res}(F, F')$.
2. Déterminer le premier entier τ tel que $p^\tau > n2^n\|F\|_2$.
3. Déterminer le premier entier σ tel que $p^\sigma > 2\theta^n\|F\|_2^{n-1}$, avec $\theta = \sqrt{n}\left(\left(1+\frac{\sqrt{n}}{2}\right)2^{(m-1)/2}(1+\eta)\eta+\frac{1}{2}\right)p^\tau$.
4. Calculer les facteurs unitaires irréductibles $f_1,\dots,f_s$ de F dans $\mathbb{Z}/p\mathbb{Z}[X]$.
5. Remonter la factorisation irréductible $f_1,\dots,f_s$ de F dans $\mathbb{Z}/p^\sigma\mathbb{Z}[X]$ à l'aide de l'Algorithme 21.2.
6. Construire la matrice (21.1) définissant le réseau L.
7. Utiliser l'Algorithme 20.2 du chapitre précédent pour obtenir une base réduite notée $w_1,\dots,w_m$, avec $m = s+n$.
8. Calculer $t = \min\{i \mid \|v_j^*\|_2 > \eta,\ \forall j > i\}$, puis la base échelonnée sur $\mathbb{Q}$ de $(w_{1,1},\dots,w_{1,s}),\dots,(w_{t,1},\dots,w_{t,s})$, qui n'est autre que $\mu_1,\dots,\mu_r$ à une permutation près.
9. Pour chaque $i \in \{1,\dots,r\}$, calculer F_i comme la partie primitive de la préimage (signée) dans $\mathbb{Z}[X]$ de $\mathrm{ct}(F)f_1^{\mu_{i,1}}\cdots f_s^{\mu_{i,s}}$.
10. Renvoyer $F_1,\dots,F_r$.

Algorithme 21.4 – Algorithme de factorisation de van Hoeij.

certains facteurs p-adiques sont déjà des projections de facteurs rationnels. On peut en plus éventuellement tester tous les produits de deux facteurs p-adiques.

La borne σ pour la précision nécessaire pour que l'algorithme fonctionne est souvent surévaluée en pratique. Il est donc utile de commencer avec une precision $\approx 2\tau$, de tester si l'on trouve des facteurs, et, si non, de doubler la précision et recommencer la réduction de réseau. On peut prouver le critère d'arrêt suivant : chaque vecteur de la base échelonnée de l'étape (8) ne contient que des 0 et 1 et conduit à un diviseur de F.

Enfin, on peut essayer de ne garder que quelques colonnes sur les n dernières, de sorte à accélérer la réduction de réseau. Dès qu'une factorisation de F est trouvée on peut ensuite redémarrer l'algorithme sur chacun des facteurs.

Exemple 21.5 Prenons pour F le produit des polynômes de Swinnerton–Dyer d'indices 2 et 3 :

$$F(X) = X^{12} - 50X^{10} + 753X^8 - 4520X^6 + 10528X^4 - 6720X^2 + 576.$$

Le plus petit nombre premier p qui ne divise pas le discriminant de F est $p = 11$. La factorisation de F modulo p est

$$(X^2+2X+10)(X^2+7X+2)(X^2+10X+1)(X^2+X+1)(X^2+9X+10)(X^2+4X+2).$$

On a donc $s = 6$, $\tau = 9$, $\eta \approx 10,7$, $\theta \approx 10^{15}$ et $\sigma = 217$. Après reduction du réseau L, on a $\|w_1^*\| \approx 1,4$, $\|w_2^*\| \approx 2,0$, et les autres $\|w_i^*\|$ dépassent 10^{108}. On a alors $t = 2$ et

$$\mu_1 = (1,0,1,0,1,1), \quad \mu_2 = (0,1,0,1,0,0),$$

$$F_1(X) = X^8 - 40X^6 + 352X^4 - 960X^2 + 576, \quad F_2(X) = X^4 - 10X^2 + 1.$$

Sur cet exemple, dès que l'on prend $\sigma \geq 12$, on trouve $t = 2$, et on vérifie que la base échelonnée de l'étape (8) contient uniquement des 0 et 1 et conduit à des facteurs de F, ce qui est suffisant pour justifier que l'on a bien trouvé tous les facteurs.

Notes

La première preuve de la borne de Landau–Mignotte est due à Landau en 1905, où elle a été donnée pour le produit des zéros de fonctions analytiques proches de zéro [Lan05]. Mignotte l'a redécouverte en 1974 pour des applications à la factorisation [Mig74].

L'inégalité de Mahler se trouve dans son article de 1961 [Mah61]. La preuve que nous en donnons ici est inspirée d'un article de Vaaler et Boyd [VB91a].

Le Lemme 21.13 est adapté d'une preuve d'Erdős du postulat de Bertrand [Erd32]. Il peut être raffiné [HW08, Chapitre 22] en

$$\prod_{\substack{2 \leq p \leq k \\ p \text{ premier}}} p = \exp((1 + o(1))k).$$

En 2001, Gerhard [Ger01] a montré que le coût binaire de la factorisation sans carré en degré n pouvait être abaissé à $\tilde{O}(n(n + \|F\|_\infty))$ en moyenne, avec un algorithme effectuant des choix aléatoires. Plus généralement, l'approche multimodulaire pour le calcul de la factorisation séparable est traitée dans l'article de Lecerf de 2008 [Lec08].

La construction des nombres p-adiques remonte aux travaux de Hensel en 1918. L'Algorithme 21.3 naïf a été popularisé en calcul formel par Zassenhaus à la fin des années soixante [Zas69]. Une implémentation comportant de nombreuses optimisations a été réalisée par Abbott, Shoup et Zimmermann [ASZ00].

Il existe des algorithmes classiques permettant d'énumérer tous les sous-ensembles d'une taille donnée les uns après les autres sans avoir besoin de les stocker tous en mémoire [NW78].

Le temps polynomial pour la factorisation des polynômes de $\mathbb{Q}[X]$ a été obtenu par A. Lenstra, H. Lenstra et L. Lovász en 1982 [LLL82], avec une borne de complexité binaire $O(n^{12} + n^9 \log^3 \|F\|_\infty)$. Schönhage a ensuite obtenu la complexité $O(n^6 + n^4 \log^2 \|F\|_\infty)$, en calculant les polynômes minimaux des racines complexes de F [Sch84]. Néanmoins, les coûts pratiques de ces algorithmes sont très élevés, et ils n'ont jamais été vraiment compétitifs par rapport à l'approche naïve.

L'algorithme de van Hoeij remonte à 2002 [Hoe02a] : bien que la borne de complexité n'était pas prouvée, une première implémentation permettait de factoriser des polynômes qui étaient auparavant hors de portée. La preuve de la complexité polynomiale est due à Belabas, van Hoeij, Klüners et Steel [Bel+09]. La présentation

qui en est faite dans ce chapitre suit cet article, qui arrive à une borne de complexité $\tilde{O}(n^8+n^6\log^2\|F\|_\infty)$, en utilisant un algorithme de réduction de réseau plus performant que celui du chapitre précédent.

Plusieurs travaux ont été poursuivis pour prouver une borne de complexité $\tilde{O}(s^6 + P(s, \log\|F\|_\infty))$ dans le pire cas, où P est un polynôme de degré total 5, qui se rapproche des coûts observés en pratique : tout d'abord dans la thèse de doctorat de Novocin [Nov08] puis dans ses articles en collaboration avec Hart, van Hoeij, Stéhlé et Villard [HHN11 ; NH09 ; NSV11].

Bibliographie

ASZ00 Abbott, John, Victor Shoup et Paul Zimmermann (2000). « Factorization in $\mathbb{Z}[x]$: the searching phase ». In : *ISSAC'00 : International Symposium on Symbolic and Algebraic Computation*. St. Andrews, Scotland : ACM Press, p. 1–7.

Bel+09 Belabas, Karim, Mark van Hoeij, Jürgen Klüners et Allan Steel (2009). « Factoring polynomials over global fields ». In : *Journal de Théorie des Nombres de Bordeaux*, vol. 21, n°1, p. 15–39.

Erd32 Erdős, Paul (1932). « Beweis eines Satzes von Tschebyschef ». In : *Acta Litterarum ac Scientiarum, Szeged*, n°5, p. 194–198.

Ger01 Gerhard, Jürgen (2001). « Fast Modular Algorithms for Squarefree Factorization and Hermite Integration ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 11, n°3, p. 203–226.

HHN11 Hart, William, Mark van Hoeij et Andrew Novocin (2011). « Practical polynomial factoring in polynomial time ». In : *ISSAC'11 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 163–170.

Hoe02a Hoeij, Mark van (2002). « Factoring polynomials and the knapsack problem ». In : *Journal of Number Theory*, vol. 95, n°2, p. 167–189.

HW08 Hardy, G. H. et E. M. Wright (2008). *An introduction to the theory of numbers*. 6e éd. Oxford University Press.

Lan05 Landau, E. (1905). « Sur quelques théorèmes de M. Petrovitch relatifs aux zéros des fonctions analytiques ». In : *Bulletin de la S. M. F.* vol. 33, p. 251–261.

Lec08 Lecerf, Grégoire (2008). « Fast separable factorization and applications ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 19, n°2, p. 135–160.

LLL82 Lenstra, A. K., H. W. Lenstra Jr. et L. Lovász (1982). « Factoring polynomials with rational coefficients ». In : *Mathematische Annalen*, vol. 261, n°4, p. 515–534.

Mah61 Mahler, K. (1961). « On the zeros of the derivative of a polynomial ». In : *Proceedings of the Royal Society of London A : Mathematical, Physical and Engineering Sciences*, vol. 264, n°1317, p. 145–154.

Mig74 Mignotte, M. (1974). « An inequality about factors of polynomials ». In : *Mathematics of Computation*, vol. 28, p. 1153–1157.

NH09 Novocin, Andy et Mark van Hoeij (2009). « Factoring univariate polynomials over the rationals ». In : *ACM Communications in Computer Algebra*, vol. 42, n°3, p. 157–157.

Nov08 Novocin, Andrew (2008). « Factoring univariate polynomials over the rationals ». Thèse de doctorat. Department of Mathematics, Florida State University, Tallahassee.

NSV11 Novocin, Andrew, Damien Stehlé et Gilles Villard (2011). « An LLL-reduction algorithm with quasi-linear time complexity ». In : *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*. STOC '11. Extended abstract. New York, NY, USA : ACM, p. 403–412.

NW78 Nijenhuis, Albert et Herbert S. Wilf (1978). *Combinatorial algorithms for computers and calculators*. Academic Press.

Sch84 Schönhage, Arnold (1984). « Factorization of univariate integer polynomials by diophantine approximation and an improved basis reduction algorithm ». In : *Automata, Languages and Programming : 11th Colloquium Antwerp, Belgium, July 16–20, 1984*. Éd. par Jan Paredaens. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 436–447.

VB91a Vaaler, Jeff et David W. Boyd (1991). « 6613, Mahler's inequality ». In : *The American Mathematical Monthly*, vol. 98, n°5, p. 451–452.

Zas69 Zassenhaus, Hans (1969). « On Hensel factorization I ». In : *Journal of Number Theory*, vol. 1, n°1, p. 291–311.

22. Factorisation des polynômes à plusieurs variables

Résumé

La factorisation des polynômes à plusieurs variables réutilise les techniques des chapitres précédents : des bornes sur les degrés partiels des facteurs, l'algorithme de Newton–Hensel,et la résolution des systèmes linéaires. Nous commençons par présenter des algorithmes généraux puis quelques variantes plus efficaces dans des cas particuliers. Ensuite nous expliquons comment ramener le cas de plus de trois variables à deux variables seulement.

Tout au long de ce chapitre $\mathbb{K}$ représente un corps effectif pour lequel on dispose d'un algorithme de factorisation dans $\mathbb{K}[X]$, et F désigne le polynôme de $\mathbb{K}[X_1,\ldots,X_n]$ qu'on souhaite factoriser. Son degré partiel en X_n est noté d, et son degré total en les seules variables $X_1,\ldots,X_{n-1}$ est noté $\eta = \deg_{X_1,\ldots,X_{n-1}}(F)$.

Il est souvent utile de voir F comme le polynôme $\tilde{F}$ en la variable X_n sur l'anneau des coefficients $\mathbb{K}[X_1,\ldots,X_{n-1}]$. Le coefficient de tête en X_n de F, noté $\mathrm{ct}_{X_n}(F)$, correspond au coefficient de tête de $\tilde{F}$. Le *contenu* de F en X_n est alors défini comme le contenu de $\tilde{F}$, c'est-à-dire le pgcd des coefficients de $\tilde{F}$. La *partie primitive* de F en X_n est la partie primitive de $\tilde{F}$, c'est-à-dire le quotient de F par son contenu en X_n. Nous dirons aussi que F est séparable en X_n lorsque $\tilde{F}$ est séparable.

Le *lemme de Gauss* (Proposition 21.1) affirmant que le contenu d'un produit de polynômes est égal au produit de leurs contenus, a pour conséquence pratique de réduire la factorisation de $\mathbb{K}[X_1,\ldots,X_n]$ à celle d'un contenu en X_n et celle d'un polynôme primitif en X_n (comme prouvé dans le Théorème 21.2). Attendu que la factorisation du contenu peut se faire par récurrence sur le nombre de variables, nous supposerons tout au long de ce chapitre que F est primitif en X_n. Ensuite, en utilisant les algorithmes du Chapitre 18, nous pouvons supposer de plus que F est séparable en X_n.

22.1 Algorithme naïf par remontée

Posons $\Delta_{X_n}(F)$, le résultant en X_n suivant :

$$\Delta_{X_n}(F) := \mathrm{Res}_{X_n}\left(F, \frac{\partial F}{\partial X_n}\right) \in \mathbb{K}[X_1, \ldots, X_{n-1}].$$

Nous commençons par expliquer comment se ramener au cas où $\Delta_{X_n}(F)$ ne s'annule pas à l'origine.

Factorisation locale

Le lemme suivant permet de vérifier que $\Delta_{X_n}(F)$ ne s'annule pas à l'origine, sans avoir à calculer $\Delta_{X_n}(F)$ explicitement.

Lemme 22.1 La condition $\Delta_{X_n}(F)(0,\ldots,0) \neq 0$ est équivalente à

$$\mathrm{ct}_{X_n}(F)(0,\ldots,0) \neq 0 \text{ et } \mathrm{Res}_{X_n}\left(F(0,\ldots,0,X_n), \frac{\partial F}{\partial X_n}(0,\ldots,0,X_n)\right) \neq 0.$$

Démonstration. La preuve est analogue à celle de Lemme 21.14 du chapitre précédent. ■

Pour se ramener au cas où $\Delta_{X_n}(F)$ ne s'annule pas à l'origine, il suffit de chercher un point $(a_1,\ldots,a_{n-1})$ dans $\mathbb{K}^{n-1}$ tel que $\mathrm{ct}_{X_n}(F)(a_1,\ldots,a_{n-1}) \neq 0$ et

$$\mathrm{Res}_{X_n}\left(F(a_1,\ldots,a_{n-1},X_n), \frac{\partial F}{\partial X_n}(a_1,\ldots,a_{n-1},X_n)\right) \neq 0,$$

puis de remplacer F par $F(X_1 + a_1, \ldots, X_{n-1} + a_{n-1}, X_n)$. Si jamais $\mathbb{K}$ est un corps fini de cardinal trop petit pour trouver un tel point, alors on peut factoriser F dans une extension algébrique suffisamment grande de $\mathbb{K}$ et nous verrons à la fin du chapitre comment en déduire la factorisation sur $\mathbb{K}$.

À partir de maintenant nous supposons que $\Delta_{X_n}(F)$ ne s'annule pas à l'origine, et nous notons $f_1, \ldots, f_s$ les facteurs irréductibles unitaires de $F(0,\ldots,0,X_n)$. Du fait que $F(0,\ldots,0,X_n)$ est séparable de même degré que F en X_n, la méthode de Newton–Hensel vue au chapitre précédent nous permet de calculer la factorisation irréductible de F dans $\mathbb{K}[[X_1,\ldots,X_n]][X_n]$ à n'importe quelle précision souhaitée. Nous noterons $F_1, \ldots, F_s$ les facteurs unitaires de F dans $\mathbb{K}[[X_1,\ldots,X_n]][X_n]$, dis *locaux*, calculés à une précision notée σ, que nous préciserons le moment venu, avec la convention que $F_i(0,\ldots,0,X_n)$ coïncide avec $f_i(X_n)$.

Recherche exhaustive des recombinaisons

Si $\sigma \geq \eta + 1$, alors chaque diviseur G de F s'écrit comme le produit

$$\mathrm{ct}_{X_n}(G)\, F_1^{\ell_1} \cdots F_s^{\ell_s} + O((X_1,\ldots,X_{n-1})^\sigma),$$

où les ℓ_i valent 0 ou 1. On pourrait supposer que la factorisation de $\mathrm{ct}_{X_n}(F)$ est disponible, par récurrence sur le nombre de variables. Cela nous conduirait directement à

Entrée $F \in \mathbb{K}[X_1,\dots,X_n]$, primitif en X_n, tel que $F(0,\dots,0,X_n)$ soit séparable de degré $\deg_{X_n}(F)$.
Sortie Les facteurs irréductibles de F.

1. Calculer les facteurs unitaires irréductibles $f_1,\dots,f_s$ de $F(0,\dots,0,X_n)$ dans $\mathbb{K}[X_n]$.
2. Poser $\sigma = 1 + \deg_{X_1,\dots,X_{n-1}}(F)$, et remonter la factorisation irréductible $f_1,\dots,f_s$ de F dans $\mathbb{K}[[X_1,\dots,X_{n-1}]][X_n]$, à la précision $(X_1,\dots,X_{n-1})^\sigma$ à l'aide de l'Algorithme 21.2 du chapitre précédent.
3. Initialiser S avec $\{1,\dots,s\}$, puis T avec $\{\}$, et L avec la liste vide.
4. Parcourir les sous-ensembles E non vides de S par taille croissante au plus $(s - \mathrm{card}(T))/2$ et disjoints de T :
   a. calculer $\mathrm{ct}_{X_n}(F)\prod_{e\in E} f_e$ à la précision $(X_1,\dots,X_{n-1})^\sigma$, puis sa préimage H dans $\mathbb{K}[X_1,\dots,X_n]$;
   b. calculer la partie primitive G de H en X_n ;
   c. si G divise F alors l'insérer dans L, remplacer T par $T \cup E$, et F par F/G.
5. Si $\deg(F) \geq 1$ alors insérer F dans L.
6. Renvoyer L.

Algorithme 22.1 – Méthode naïve de factorisation par remontée et recombinaison.

un algorithme énumérant tous les vecteurs possibles pour $(\ell_1,\dots,\ell_s)$ et tous les diviseurs de $\mathrm{ct}_{X_n}(F)$. Néanmoins, pour limiter le coût exponentiel induit par le parcours des diviseurs de $\mathrm{ct}_{X_n}(F)$, nous préférons nous y prendre différemment. Comme on a $\deg_{X_1,\dots,X_{n-1}}(G) \leq \deg_{X_1,\dots,X_{n-1}}(F)$ et comme $\mathrm{ct}_{X_n}(F)/\mathrm{ct}_{X_n}(G)\,G$ est aussi un diviseur de F, alors on a la borne un peu plus fine

$$\deg_{X_1,\dots,X_{n-1}}(\mathrm{ct}_{X_n}(F)/\mathrm{ct}_{X_n}(G)\,G) \leq \eta.$$

Par conséquent, $\mathrm{ct}_{X_n}(F)\,f_1^{\ell_1}\cdots f_s^{\ell_s}$ est un polynôme de degré total au plus η en les variables $X_1,\dots,X_{n-1}$.

À l'inverse, si l'on nous donne un vecteur $(\ell_1,\dots,\ell_s) \in \{0,1\}^s$, nous pouvons d'abord construire le polynôme $H \in \mathbb{K}[X_1,\dots,X_n]$ comme la préimage de

$$\mathrm{ct}_{X_n}(F)\,f_1^{\ell_1}\cdots f_s^{\ell_s},$$

qui ne conserve que les termes en degré total au plus η en $X_1,\dots,X_{n-1}$. Ensuite, on calcule la partie primitive G de H en X_n. Enfin, on teste si G divise F. Le choix de la précision $\sigma \geq \eta + 1$ nous assure que nous pouvons découvrir tous les facteurs irréductibles de F en testant tous les vecteurs $(\ell_1,\dots,\ell_s)$ de $\{0,1\}^s$. Cette approche est résumée dans l'Algorithme 22.1 (à comparer avec l'Algorithme 21.3 en page 388).

Définir des structures de données pour représenter les polynômes à plusieurs variables et analyser les coûts des opérations arithmétiques, nous éloignerait trop des

objectifs de cet ouvrage. Dans ce chapitre, nous analyserons le coût des algorithmes de factorisation dans le seul cas où $n = 2$.

Proposition 22.2 L'Algorithme 22.1 est correct. Si $n = 2$, il effectue une factorisation irréductible de degré d dans $\mathbb{K}[X_2]$, plus $\tilde{O}(\eta 2^d)$ opérations arithmétiques dans $\mathbb{K}$.

Démonstration. Nous venons déjà de justifier que tous les polynômes insérés dans la liste L sont des diviseurs de F. Le fait de parcourir les sous-ensembles E par taille croissante assure que les facteurs trouvés sont irréductibles. Enfin, quand on arrive à l'étape (5), cela signifie qu'aucun facteur de la valeur courante de F n'est obtenu à partir d'au plus la moitié de ses facteurs locaux. Par conséquent, un tel polynôme F est soit constant soit irréductible. Par ces faits, nous déduisons que l'algorithme est correct.

La première étape correspond à la factorisation à une variable de l'énoncé de la proposition. Le coût de la remontée de Newton–Hensel de l'étape (2) est quasi-linéaire, par la Proposition 21.20 du chapitre précédent.

Pour le calcul de H, on construit l'arbre des sous-produits présenté dans la Section 5.4, de sorte que la Proposition 5.5 implique un coût quasi-linéaire. Pour le calcul du contenu de H on peut utiliser successivement d fois l'algorithme rapide de pgcd, pour obtenir aussi un coût quasi-linéaire. Pour tester si G divise F on peut calculer F/G dans $\mathbb{K}[[X_1,\dots,X_n]][X_n]$ à la précision $(X_1,\dots,X_{n-1})^\sigma$ puis la préimage Q de ce quotient dans $\mathbb{K}[X_1,\dots,X_n]$ et enfin vérifier que F est égal à GQ. On obtient ainsi un coût quasi-linéaire.

Enfin, le nombre de candidats G à construire et tester dans l'étape (6), tant que F et S ne changent pas, est borné par le nombre de sous-ensembles de $S \setminus T$ de taille au plus $(s - \mathrm{card}(T))/2$, c'est-à-dire $2^{s-\mathrm{card}(T)-1}$. ∎

Exemple 22.1 Prenons $\mathbb{K} = \mathbb{F}_p$ avec $p = 101$, et

$$F(X_1,X_2) = X_2^4 + 99X_2^3 + 100X_2^2 + (2X_1^2+2)X_2 + 100X_1^4 + 100X_1^2.$$

Le polynôme $F(0,X_2) = X_2^4 + 99X_2^3 + 100X_2^2 + 2X_2$ est séparable. On trouve $d = 4$, $\eta = 4$, et $s = 4$. Les facteurs locaux à la précision X_1^5 sont :

$$\begin{aligned}
\mathsf{F}_1 &= X_2 + 50X_1^2 + 63X_1^4 + O(X_1^5),\\
\mathsf{F}_2 &= X_2 + 99 + 51X_1^2 + 38X_1^4 + O(X_1^5),\\
\mathsf{F}_3 &= X_2 + 100 + 50X_1^2 + 38X_1^4 + O(X_1^5),\\
\mathsf{F}_4 &= X_2 + 1 + 51X_1^2 + 63X_1^4 + O(X_1^5).
\end{aligned}$$

L'algorithme commence par tester si l'une des préimages des F_i divise F, ce qui n'est pas le cas. Il parcourt ensuite les sous-ensembles $E = \{i,j\}$ de taille 2 de $\{1,2,3,4\}$ et teste si l'une des préimages de $\mathsf{F}_i\mathsf{F}_j$ divise F. On trouve juste $\{1,2\}$ et $\{3,4\}$, et on obtient $r = 2$ facteurs irréductibles $F_1 = X_2^2 + 99X_2 + X_1^2$ et $F_2 = X_2^2 + 100X_1^2 + 100$.

Les optimisations vues dans la Section 21.3 du chapitre précédent s'adaptent aisément ici.

22.2 Recombinaison des dérivées logarithmiques

Nous présentons dans cette section un algorithme de factorisation fonctionnant en temps polynomial, sous les mêmes hypothèses que dans la section précédente. Et comme pour l'algorithme précédent, nous commençons par calculer une factorisation locale de F en X_n, mais jusqu'à une précision plus élevée de sorte à pouvoir résoudre le problème de recombinaison au moyen d'algèbre linéaire. L'idée clé est la même que dans le chapitre précédent : si F se factorise en $F_1 \cdots F_r$ alors le calcul des dérivées logarithmiques en X_n conduit à

$$\frac{\partial F}{\partial X_n} = \frac{F}{F_1}\frac{\partial F_1}{\partial X_n} + \cdots + \frac{F}{F_s}\frac{\partial F_s}{\partial X_n},$$

ce qui permet, d'une certaine façon, de « linéariser » le problème.

Dérivées logarithmiques

Nous continuons de noter $f_1, \ldots, f_s$ les facteurs locaux unitaires de F en précision $(X_1, \ldots, X_{n-1})^\sigma$. Pour chaque facteur irréductible F_i de F on définit le vecteur $\mu_i \in \{0,1\}^s$ par l'égalité

$$F_i = \mathrm{ct}_{X_n}(F_i)\, f_1^{\mu_{i,1}} \cdots f_s^{\mu_{i,s}} + O((X_1, \ldots, X_{n-1})^\sigma).$$

En prenant la dérivée logarithmique et en multipliant par F les deux membres de cette égalité, nous obtenons :

$$\frac{F}{F_i}\frac{\partial F_i}{\partial X_n} = \mu_{i,1}\frac{F}{f_1}\frac{\partial f_1}{\partial X_n} + \cdots + \mu_{i,s}\frac{F}{f_s}\frac{\partial f_s}{\partial X_n} + O((X_1, \ldots, X_{n-1})^{\sigma-1}).$$

Si G est un diviseur primitif de F qui s'écrit

$$G = \mathrm{ct}_{X_n}(G)\, f_1^{\ell_1} \cdots f_s^{\ell_s} + O((X_1, \ldots, X_{n-1})^\sigma),$$

avec $(\ell_1, \ldots, \ell_s) \in \{0,1\}^s$, alors $\frac{F}{G}\frac{\partial G}{\partial X_n}$ est de degré total en $X_1, \ldots, X_{n-1}$ au plus η et on a

$$\frac{F}{G}\frac{\partial G}{\partial X_n} = \ell_1\frac{F}{f_1}\frac{\partial f_1}{\partial X_n} + \cdots + \ell_s\frac{F}{f_s}\frac{\partial f_s}{\partial X_n} + O((X_1, \ldots, X_{n-1})^{\sigma-1}).$$

Observons que $(\ell_1, \ldots, \ell_s)$ est la somme d'un sous-ensemble des μ_i. On obtient ainsi la propriété suivante satisfaite par les ℓ_i :

$$\ell_1\frac{F}{f_1}\frac{\partial f_1}{\partial X_n} + \cdots + \ell_s\frac{F}{f_s}\frac{\partial f_s}{\partial X_n}$$

est un polynôme de degré total au plus η en $X_1, \ldots, X_{n-1}$, modulo $(X_1, \ldots, X_{n-1})^\sigma$. Nous allons montrer que cette condition est suffisante pour trouver des diviseurs de F lorsque σ est suffisamment grand.

Troncatures

Si $\mathfrak{a}$ est une série entière, nous notons $\lceil \mathfrak{a} \rceil^\sigma$ le polynôme obtenu à partir des seuls termes de $\mathfrak{a}$ de degrés totaux au plus $\sigma - 1$. On définit ensuite $\lfloor \mathfrak{a} \rfloor_\tau$ comme la série obtenue supprimant tous les termes de $\mathfrak{a}$ de degrés totaux au plus $\tau - 1$ et on note

$$[\mathfrak{a}]_\tau^\sigma = \lceil \lfloor \mathfrak{a} \rfloor_\tau \rceil^\sigma.$$

Contrairement au chapitre précédent, la fonction $[\cdot]_\tau^\sigma$ est linéaire : si $\mathfrak{a}_1, \dots, \mathfrak{a}_s$ sont des séries, et si $\ell_1, \dots, \ell_s$ sont dans $\mathbb{K}$, alors on peut écrire

$$[\ell_1 \mathfrak{a}_1 + \cdots + \ell_s \mathfrak{a}_s]_\tau^\sigma = \ell_1 [\mathfrak{a}_1]_\tau^\sigma + \cdots + \ell_s [\mathfrak{a}_s]_\tau^\sigma + O((X_1, \dots, X_{n-1})^\sigma).$$

Si $\mathfrak{P} = \mathfrak{P}_0 + \mathfrak{P}_1 X_n + \cdots + \mathfrak{P}_m X_n^m$ est un polynôme de $\mathbb{K}[[X_1, \dots, X_{n-1}]][X_n]$, on étend naturellement la notation coefficient par coefficient comme suit :

$$[\mathfrak{P}]_\tau^\sigma = [\mathfrak{P}_0]_\tau^\sigma + [\mathfrak{P}_1]_\tau^\sigma X_n + \cdots + [\mathfrak{P}_m]_\tau^\sigma X_n^m.$$

On introduit les polynômes

$$\mathfrak{G}_i = \left[\frac{F}{F_i} \frac{\partial F_i}{\partial X_n} \right]_{\eta+1}^{\sigma-1},$$

puis le système linéaire L en les inconnues $\ell_1, \dots, \ell_n$ dans $\mathbb{K}$ dont les equations sont :

$$\ell_1 \mathfrak{G}_1 + \cdots + \ell_s \mathfrak{G}_s = 0. \tag{22.1}$$

Proposition 22.3 Si $\sigma \geq 2 + \eta(2d-1)$ alors $\mu_1, \dots, \mu_r$ est une base des solutions de L.

Démonstration. Par construction, les μ_i sont bien des solutions du système. Notons $\ell = (\ell_1, \dots, \ell_n)$ une solution non nulle de L. Notons j un indice tel que $\ell_j \neq 0$, et k l'indice tel que $\mu_{k,j} = 1$. On définit $\tilde{\ell} = \ell - \ell_j \mu_k$, qui satisfait $\tilde{\ell}_j = 0$. On considère ensuite

$$\mathfrak{H} = \tilde{\ell}_1 \frac{F}{F_1} \frac{\partial F_1}{\partial X_n} + \cdots + \tilde{\ell}_s \frac{F}{F_s} \frac{\partial F_s}{\partial X_n} + O((X_1, \dots, X_{n-1})^{\sigma-1}),$$

puis $H = \lceil \mathfrak{H} \rceil^{\eta+1}$. Par construction on a alors : $\mathfrak{H} = H + O((X_1, \dots, X_{n-1})^{\sigma-1})$ et F_j divise H à la précision $O((X_1, \dots, X_{n-1})^{\sigma-1})$.

Par la multiplicativité du résultant, le polynôme

$$R = \mathrm{Res}_{X_n}(F, H) \in \mathbb{K}[X_1, \dots, X_{n-1}]$$

s'annule donc à la précision $O((X_1, \dots, X_{n-1})^{\sigma-1})$. Comme son degré total est au plus $\eta(2d-1)$, l'hypothèse sur la précision σ implique que R est le polynôme nul. Par conséquent, le polynôme F_k, qui est multiple de F_j, divise H, ce qui prouve que $\tilde{\ell}_l = 0$ pour tout indice l tels que $\mu_{k,l} = 1$.

En procédant de même avec les autres indices j de ℓ, nous pouvons déduire que ℓ est bien dans l'espace des μ_i, ce qui conclut la preuve. ∎

Entrée $F \in \mathbb{K}[X_1,\dots,X_n]$, primitif et tel que $F(0,\dots,0,X_n)$ soit séparable en X_n de degré $d = \deg_{X_n}(F)$.
Sortie Les facteurs irréductibles de F.

1. Calculer les facteurs unitaires irréductibles $f_1,\dots,f_s$ de $F(0,\dots,0,X_n)$ dans $\mathbb{K}[X_n]$.
2. Poser $\eta = \deg_{X_1,\dots,X_{n-1}}(F)$, puis $\sigma = 2 + \eta(2d-1)$, et remonter la factorisation irréductible $f_1,\dots,f_s$ de F dans $\mathbb{K}[[X_1,\dots,X_{n-1}]][X_n]$ à la précision $(X_1,\dots,X_{n-1})^\sigma$ à l'aide de l'Algorithme 21.2 du chapitre précédent.
3. Construire la matrice L définie dans (22.1).
4. Calculer la base échelonnée réduite des solutions de L, qui n'est autre que $\mu_1,\dots,\mu_r$.
5. Pour chaque $i \in \{1,\dots,r\}$, calculer F_i comme la partie primitive de $\lceil \mathrm{ct}_{X_n}(F)\, F_1^{\mu_{i,1}} \cdots F_s^{\mu_{i,s}} \rceil^{\eta+1}$.
6. Renvoyer $F_1,\dots,F_r$.

Algorithme 22.2 – Algorithme rapide de factorisation à plusieurs variables.

Synthèse de l'algorithme

La méthode de factorisation basée sur les idées que nous venons de présenter est résumée dans l'Algorithme 22.2. Nous notons θ un exposant réalisable pour la complexité du produit des matrices au sens du Chapitre 8 : deux matrices de taille $n \times n$ peuvent être multipliées en temps $O(n^\theta)$.

Théorème 22.4 L'Algorithme 22.2 est correct. Si $n = 2$, il nécessite de factoriser un polynôme de degré d dans $\mathbb{K}[X_2]$, plus $\tilde{O}(\eta d^{\theta+1})$ opérations arithmétiques dans $\mathbb{K}$.

Démonstration. La preuve de l'algorithme découle de la Proposition 22.3 qui garantit l'assertion de l'étape (4). Supposons maintenant $n = 2$, et analysons la complexité. L'étape (2) coûte $\tilde{O}(\eta d^2)$ opérations dans $\mathbb{K}$, par la Proposition 21.20 du Chapitre précédent. Le calcul de chaque $\mathfrak{G}_i$ peut se faire en temps $\tilde{O}(\eta d s)$ par le lemme suivant. Par conséquent le coût de l'étape (3) est $\tilde{O}(\eta d^3)$.

Le système linéaire comporte $O(d)$ inconnues et $O(\eta d^2)$ équations, une base des solutions peut être calculée en temps $\tilde{O}(\eta d^{\theta+1})$. Comme les μ_i forment une base orthogonale très particulière, une base échelonnée des solutions de L peut être déduite en temps linéaire.

La reconstruction des facteurs à l'étape (5) est réalisable en temps quasi-linéaire en utilisant les arbres des sous-produits. ■

Exemple 22.2 Avec le polynôme F de l'Exemple 22.1, nous obtenons

$$\begin{aligned}
\frac{F}{F_1}\frac{\partial F_1}{\partial X_n} &= X_2^3 + (99 + 51X_1^2 + 38X_1^4 + 19X_1^6 + 75X_1^8 + O(X_1^{10}))X_2^2 \\
&\quad + (100 + 100X_1^2 + O(X_1^{10}))X_2 \\
&\quad + 2 + 52X_1^2 + 12X_1^4 + 44X_1^6 + 7X_1^8 + O(X_1^{10}), \\
\frac{F}{F_2}\frac{\partial F_2}{\partial X_n} &= X_2^3 + (50X_1^2 + 63X_1^4 + 82X_1^6 + 26X_1^8 + O(X_1^{10}))X_2^2 \\
&\quad + (100 + 100X_1^2 + O(X_1^{10}))X_2 \\
&\quad + 51X_1^2 + 89X_1^4 + 57X_1^6 + 94X_1^8 + O(X_1^{10}), \\
\frac{F}{F_3}\frac{\partial F_3}{\partial X_n} &= X_2^3 + (100 + 51X_1^2 + 63X_1^4 + 19X_1^6 + 26X_1^8 + O(X_1^{10}))X_2^2 \\
&\quad + \left(99 + 76X_1^4 + 63X_1^6 + 49X_1^8 + O(X_1^{10})\right)X_2 \\
&\quad + X_1^2 + 51X_1^4 + 63X_1^6 + 19X_1^8 + O(X_1^{10}), \\
\frac{F}{F_4}\frac{\partial F_4}{\partial X_n} &= X_2^3 + (98 + 50X_1^2 + 38X_1^4 + 82X_1^6 + 75X_1^8 + O(X_1^{10}))X_2^2 \\
&\quad + \left(2 + 2X_1^2 + 25X_1^4 + 38X_1^6 + 52X_1^8 + O(X_1^{10})\right)X_2 \\
&\quad + 100X_1^2 + 50X_1^4 + 38X_1^6 + 82X_1^8 + O(X_1^{10}).
\end{aligned}$$

Si on prend $\sigma = 8$, alors on trouve le système linéaire

$$\begin{aligned}
44\ell_1 + 57\ell_2 + 63\ell_3 + 38\ell_4 &= 0, \\
63\ell_3 + 38\ell_4 &= 0, \\
19\ell_1 + 82\ell_2 + 19\ell_3 + 82\ell_4 &= 0.
\end{aligned}$$

La base échelonnée des solutions est $\mu_1 = (1,1,0,0)$ et $\mu_2 = (0,0,1,1)$. Avec cet exemple, on trouve donc la factorisation de F sans avoir besoin d'aller à la borne de précision $2 + \eta(2d-1) = 30$. Il est donc très utile en pratique d'essayer des précisions plus petites plutôt que d'utiliser directement la borne.

22.3 Algorithme de Lecerf

Lorsque la caractéristique p de $\mathbb{K}$ est nulle ou bien non nulle mais suffisamment grande, il est possible d'améliorer nettement la borne de précision σ de l'algorithme précédent, en changeant de système linéaire L. C'est ce que nous allons montrer dans les paragraphes suivants.

Soit G un polynôme de degré au plus $d-1$ en X_n et de degré total en les variables $X_1,\dots,X_{n-1}$ au plus η. On note $\varphi_1,\dots,\varphi_d$ les racines de F vu comme un polynôme de $\bar{\mathbb{K}}[[X_1,\dots,X_{n-1}]][X_n]$. En notant ρ_i le résidu de G/F en φ_i,

$$\rho_i = \frac{G(X_1,\dots,X_{n-1},\varphi_i)}{\frac{\partial F}{\partial X_n}(X_1,\dots,X_{n-1},\varphi_i)},$$

on a la décomposition en éléments simples de G/F suivante :

$$\frac{G}{F} = \frac{\rho_1}{X_n - \varphi_1} + \cdots + \frac{\rho_d}{X_n - \varphi_d}.$$

Pour tout indice $k \leq n-1$, la dérivée partielle de ρ_i en X_k est donnée par

$$\frac{\partial \rho_i}{\partial X_k} = \frac{N_k(X_1, \ldots, X_{n-1}, \varphi_i, \frac{\partial \varphi_i}{\partial X_k})}{\frac{\partial F}{\partial X_n}(X_1, \ldots, X_{n-1}, \varphi_i)^2},$$

où

$$N_k(X_1, \ldots, X_n, Y) = \left(\frac{\partial G}{\partial X_k} + \frac{\partial G}{\partial X_n} Y\right) \frac{\partial F}{\partial X_n} - G\left(\frac{\partial^2 F}{\partial X_k X_n} + \frac{\partial^2 F}{\partial X_n^2} Y\right).$$

Ensuite, en utilisant

$$\frac{\partial \varphi_j}{\partial X_k} = -\frac{\frac{\partial F}{\partial X_k}}{\frac{\partial F}{\partial X_n}}$$

et en posant

$$P_k(X_1, \ldots, X_n) = \left(\frac{\partial G}{\partial X_k}\frac{\partial F}{\partial X_n} - \frac{\partial G}{\partial X_n}\frac{\partial F}{\partial X_k}\right)\frac{\partial F}{\partial X_n} - G\left(\frac{\partial^2 F}{\partial X_k X_n}\frac{\partial F}{\partial X_n} - \frac{\partial^2 F}{\partial X_n^2}\frac{\partial F}{\partial X_k}\right), \quad (22.2)$$

on parvient à exprimer la dérivée du résidu de façon purement algébrique en terme de φ_i :

$$\frac{\partial \rho_i}{\partial X_k} = \frac{P_k(X_1, \ldots, X_{n-1}, \varphi_j)}{\frac{\partial F}{\partial X_n}(X_1, \ldots, X_{n-1}, \varphi_j)^3}.$$

La nullité de $\frac{\partial \rho_i}{\partial X_k}$ pour tout $i \in \{1, \ldots, s\}$ est équivalente à la divisibilité de P_k par F dans $\bar{\mathbb{K}}[[X_1, \ldots, X_{n-1}]][X_n]$. En observant que P_k est de degré total au plus 3η en $X_1, \ldots, X_{n-1}$, et de degré au plus $3d-3$ en X_n, le lemme suivant donne une borne suffisante pour la précision permettant de tester cette divisibilité.

Lemme 22.5 Soit P un polynôme de degré total au plus 3η en $X_1, \ldots, X_{n-1}$ pour $1 \leq i \leq n-1$, et de degré au plus $3d-3$ en X_n. On note Q et R le quotient et le reste de la division de P par F dans $\mathbb{K}[[X_1, \ldots, X_{n-1}]][X_n]$, de sorte que $P = QF + R$. Alors, le polynôme F divise P dans $\mathbb{K}[[X_1, \ldots, X_{n-1}]][X_n]$ si, et seulement si, $[Q]_{2\eta+1}^{3\eta+1} = 0$ et $\lceil R \rceil^{3\eta+1} = 0$.

Démonstration. Supposons que F divise P dans $\mathbb{K}[[X_1, \ldots, X_{n-1}]][X_n]$. On a alors $R = 0$ et $P = QF$, avec $Q \in \mathbb{K}(X_1, \ldots, X_{n-1})[X_n]$. Notons $Q = A/a$ avec $A \in \mathbb{K}[X_1, \ldots, X_n]$ et $a \in \mathbb{K}[X_1, \ldots, X_{n-1}]$ premiers entre eux, de sorte à pouvoir écrire $aP = AF$ dans $\mathbb{K}[X_1, \ldots, X_n]$. Comme F est primitif, le lemme de Gauss (Proposition 21.1) sur la multiplicativité des contenus assure que a est un polynôme constant. On obtient ainsi l'égalité $P = QF$ dans $\mathbb{K}[X_1, \ldots, X_n]$. Comme le degré total de Q en $X_1, \ldots, X_{n-1}$ est au plus 2η, on déduit $[Q]_{2\eta+1}^{3\eta+1} = 0$.

Réciproquement, supposons $\lceil R \rceil^{3\eta+1} = 0$ et $[Q]_{2\eta+1}^{3\eta+1} = 0$. Comme le degré total de $\lceil Q \rceil^{2\eta+1} F$ en $X_1, \ldots, X_{n-1}$ est au plus 3η, on a l'égalité $P = \lceil Q \rceil^{2\eta+1} F$ dans $\mathbb{K}[X_1, \ldots, X_n]$, et donc F divise P dans $\mathbb{K}[[X_1, \ldots, X_{n-1}]][X_n]$. ■

Pour chaque $1 \leq i \leq s$ et chaque $1 \leq k \leq n-1$ on introduit

$$\mathfrak{G}_{i,k} = \left\lceil \frac{F}{f_i} \frac{\partial f_i}{\partial X_n} \right\rceil^{\eta+1}.$$

On note $P_{i,k}$ la valeur de P_k obtenue lorsqu'on remplace G par $\mathfrak{G}_{i,k}$ dans la formule (22.2). Enfin les polynômes $Q_{i,k}$ et $R_{i,k}$ représentent le quotient et le reste de la division de $P_{i,k}$ par F dans $\mathbb{K}[[X_1, \ldots, X_{n-1}]][X_n]$ à la précision $(X_1, \ldots, X_{n-1})^{3\eta+1}$. Le système linéaire L′ à considérer maintenant est formé par les équations

$$\begin{aligned} &\ell_1 [Q_{1,k}]_{2\eta+1}^{3\eta+1} + \cdots + \ell_s [Q_{s,k}]_{2\eta+1}^{3\eta+1} = 0, && \text{et} \\ &\ell_1 \lceil R_{1,k} \rceil^{3\eta+1} + \cdots + \ell_s \lceil R_{s,k} \rceil^{3\eta+1} = 0, && \text{pour } 1 \leq k \leq n-1. \end{aligned}$$

Lemme 22.6 Si $\mathbb{K}$ est de caractéristique nulle ou supérieure ou égale à $1+\eta(2d-1)$, alors les μ_i forment une base des solutions de L′, à une permutation près.

Démonstration. Indépendamment de l'hypothèse sur $\mathbb{K}$, par construction, les μ_i sont clairement des solutions de L′. Soit $\ell = (\ell_1, \ldots, \ell_s)$ un vecteur solution de L′. On considère $G = \ell_1 \mathfrak{G}_1 + \cdots + \ell_s \mathfrak{G}_s$. Par construction, les résidus de $\frac{G}{F}$ en X_n ont toutes leurs dérivées partielles nulles. Par conséquent le résidu ρ_i de G/F en φ_i satisfait

$$\rho_i = \ell_i + O((X_1, \ldots, X_{n-1})^\tau),$$

avec $\tau = \infty$ si $p = 0$ et $\tau = p$ sinon, et on obtient

$$\frac{G}{F} = \frac{\ell_1}{X - \varphi_1} + \cdots + \frac{\ell_1}{X - \varphi_s} + O((X_1, \ldots, X_{n-1})^\tau).$$

Soit j un indice tel que $\ell_j \neq 0$, et soit k l'indice tel que $\mu_{k,j} = 1$. On pose $\tilde{\ell} = \ell - \ell_j \mu_k$, de sorte que $\tilde{\ell}_j = 0$, et on introduit $\tilde{G} = G - \ell_j F_k$. Les $\tilde{\ell}_1, \ldots, \tilde{\ell}_s$ sont les résidus de $\tilde{G}/F$, et $X_n - \varphi_j$ divise G à la précision $O((X_1, \ldots, X_{n-1})^\tau)$. Le résultant $R = \mathrm{Res}_{X_n}(F, G)$ s'annule donc à cette précision.

Avec l'hypothèse sur la caractéristique, comme le degré total de R est au plus $\eta(2d-1)$, il s'ensuit que R est le polynôme nul. Par conséquent F_k divise $\tilde{G}$ et $\tilde{\ell}_l$ est nul pour tout indice l tel que $\mu_{k,l} = 1$. En procédant ainsi avec tous les indices j, on vient de prouver que $\ell_{j_1} = \ell_{j_2}$ dès lors qu'il existe k tel que $\mu_{k,j_1} = \mu_{k,j_2} = 1$. En d'autres termes, ℓ s'exprime comme combinaison linéaire des μ_i. ■

Théorème 22.7 Si $\mathbb{K}$ est de caractéristique nulle ou supérieure ou égale à $1+\eta(2d-1)$, et si $n=2$, alors la factorisation de $F \in \mathbb{K}[X_1, X_2]$ peut se faire en factorisant un polynôme de degré d dans $\mathbb{K}[X_2]$, plus $\tilde{O}(\eta d^\theta)$ opérations arithmétiques dans $\mathbb{K}$.

Démonstration. On modifie l'Algorithme 22.2 en prenant pour précision $\sigma = \eta + 1$, et en utilisant le système linéaire L′ à la place de L. Le lemme précédent assure que l'algorithme est correct.

Quant à la complexité, lorsque $n=2$, il faut compter un temps quasi-linéaire pour les étapes (2) et (5). Dans l'étape (3), le calcul de chaque $\mathfrak{G}_{i,1}$ peut être réalisé en temps quasi-linéaire. Les calculer tous requiert un coût $\tilde{O}(\eta d^2)$. Le système L′ est constitué de $O(\eta d)$ équations en $O(d)$ inconnues. Il peut donc être résolu au moyen de $O(\eta d^\theta)$ opérations dans $\mathbb{K}$ en utilisant le Théorème 8.6 en page 175. En déduire les μ_i peut se faire en temps linéaire. ■

Lorsque $n=2$ et qu'on peut échanger les rôles des deux variables, il est préférable de prendre pour deuxième variable celle de plus petit degré partiel, de sorte que la complexité du théorème puisse être bornée par $\tilde{O}((\eta d)^{(\theta+1)/2})$, ce qui s'interprète comme un coût sous-quadratique dans la taille dense de F.

22.4 Réduction probabiliste au cas de deux variables

La taille des systèmes linéaires L et L′ intervenant dans les algorithmes précédents croît rapidement avec le nombre de variables. Nous présentons ici une méthode probabiliste pour ramener la factorisation au cas de seulement deux variables.

Nous continuons à travailler sous les mêmes hypothèses : F est primitif en X_n, et $F(0,\dots,0,X_n)$ est séparable de degré $d = \deg_{X_n}(F)$. L'entier η représente le degré total de F en les $n-1$ premières variables. Bien sûr nous supposons $n \geq 3$ tout au long de cette section. Nous allons voir que pour presque toutes les valeurs de $(a_1,\dots,a_{n-1}) \in \mathbb{K}^{n-1}$, les facteurs irréductibles de F sont en bijection naturelle avec ceux de $F(a_1Y,\dots,a_{n-1}Y,Z) \in \mathbb{K}[Y,Z]$, où Y et Z sont des nouvelles variables.

On introduit de nouvelles variables $A_1,\dots,A_{n-1}$. Le lemme suivant opère un changement de variables suffisamment générique pour préserver l'irréductibilité :

Lemme 22.8 Si F est irréductible, alors le polynôme $F(A_1Y,\dots A_{n-1}Y,Z)$ est irréductible dans $\mathbb{K}(A_1,\dots,A_{n-1})[Y,Z]$.

Démonstration. Posons

$$G(A_1,\dots,A_{n-1},Y,Z) = F(A_1Y,\dots A_{n-1}Y,Z).$$

Comme G est primitif en Z et appartient à $\mathbb{K}[A_1,\dots,A_{n-1},Y,Z]$, il suffit de prouver qu'il est irréductible dans $\mathbb{K}[A_1,\dots,A_{n-1},Y,Z]$.

Comme $F(0,\dots,0,Z) = G(0,\dots,0,1,Z)$, les facteurs irréductibles unitaires $\mathfrak{F}_1,\dots,\mathfrak{F}_s$ de F vus comme polynômes de$\mathbb{K}[[X_1,\dots,X_{n-1}]][X_n]$ sont en bijection avec ceux de G dans $\mathbb{K}(A_1,\dots,A_{n-1})[[Y]][Z]$, notés $\mathfrak{G}_1,\dots,\mathfrak{G}_s$. Autrement dit, quitte à permuter les indices, on a $\mathfrak{G}_i = \mathfrak{F}_i(A_1Y,\dots,A_{n-1}Y,Z)$. Par conséquent, tout facteur irréductible de G

dans $\mathbb{K}[A_1,\dots,A_{n-1},Y,Z]$ est de la forme $H(A_1Y,\dots,A_{n-1}Y,Z)$ avec $H\in\mathbb{K}[X_1,\dots,X_n]$. En remplaçant Y par 1, il s'ensuit que le polynôme $H(A_1,\dots,A_{n-1},Z)$ est nécessairement un facteur de $F(A_1,\dots,A_{n-1},Z)$, et donc que G est irréductible. ■

Rappelons que les facteurs irréductibles de F sont notés $F_1,\dots,F_r$. Un point $(a_1,\dots,a_{n-1})$ de $\mathbb{K}^{n-1}$ est dit *de Hilbert* pour F lorsque tous les polynômes

$$F_i(a_1Y,\dots,a_{n-1}Y,Z)$$

sont irréductibles. En d'autres termes, les facteurs irréductibles de F sont en bijection avec ceux de $F(a_1Y,\dots,a_{n-1}Y,Z)$.

Théorème 22.9 Soit F un polynôme primitif en X_n, et tel que $F(0,\dots,0,X_n)$ est séparable de degré $d=\deg_{X_n}(F)$. Il existe un polynôme $\mathcal{A}\in\mathbb{K}[A_1,\dots,A_{n-1}]$ non nul de degré au plus $\eta(d-1)(2d-1)$ qui s'annule en tous les points de $\mathbb{K}^{n-1}$ qui ne sont pas de Hilbert pour F.

Démonstration. Comme la fonction $\delta(d)=(d-1)(2d-1)$ satisfait $\delta(d_1)+\delta(d_2)\le\delta(d_1+d_2)$ pour n'importe quels entiers strictement positifs d_1 et d_2, il suffit de prouver le résultat lorsque F est irréductible.

Posons $G(A_1,\dots,A_{n-1},Y,Z)=F(A_1Y,\dots A_{n-1}Y,Z)$, et réutilisons les notations de la preuve précédente. Avec $\sigma=2+\eta(2d-1)$, la Proposition 22.3 nous garantit que le système linéaire

$$\ell_1\left[\frac{F}{F_1}\frac{\partial F_1}{\partial X_1}\right]_{\eta+1}^{\sigma-1}+\cdots+\ell_s\left[\frac{F}{F_s}\frac{\partial F_s}{\partial X_s}\right]_{\eta+1}^{\sigma-1}=0$$

est de rang $s-1$. Il existe donc un mineur non nul $\mathcal{A}$ de ce système de taille $s-1$. Si maintenant $A(a_1,\dots,a_{n-1})\neq 0$, alors le système reste de rang $s-1$ lorsqu'on spécialise les A_i en les a_i. Pour une telle spécialisation, le polynôme $F(a_1Y,\dots,a_{n-1}Y,Z)$ est donc irréductible, en utilisant la Proposition 22.3.

Pour chaque $j\ge 0$ et $k\ge 0$, le coefficient de Y^jZ^k dans

$$\frac{F}{F_l}\frac{\partial F_1}{\partial X_l}$$

est de degré au plus j. Le degré total de $\mathcal{A}$ est donc au plus $(s-1)\eta(2d-1)$. ■

Lemme 22.10 — Schwartz–Zippel. Soit S un sous-ensemble fini de $\mathbb{K}$. Si $\mathcal{A}$ est un polynôme non nul de $\mathbb{K}[A_1,\dots,A_m]$, alors il a au plus $\deg(A)\,\mathrm{card}(S)^{m-1}$ racines dans S^m.

Démonstration. Lorsque $m=1$, le résultat est immédiat. Supposons par récurrence la borne prouvée pour $m\ge 1$. Le coefficient de tête $\mathrm{ct}_{A_n}(\mathcal{A})$ appartient à $\mathbb{K}[A_1,\dots,A_{m-1}]$ et est de degré total au plus $\delta=\deg(A)-\deg_{A_n}(A)$. Il admet au plus $\delta\,\mathrm{card}(S)^{m-2}$ racines dans S^{m-1}. Par ailleurs, les points qui annulent $\mathcal{A}$ sans annuler son coefficient

de tête en A_n, sont forcément au plus $\deg_{A_n}(A)\operatorname{card}(S)^{m-1}$. Au total le nombre de zéros est borné par

$$(\delta\operatorname{card}(S)^{m-2})\operatorname{card}(S)+\deg_{A_n}(A)\operatorname{card}(S)^{m-1}=\deg(A)\operatorname{card}(S)^{m-1},$$

comme annoncé. ■

On déduit naturellement de ces résultats une approche probabiliste efficace. On fixe un ensemble S de taille au moins $C\eta(d-1)(2d-1)$, pour une constante C assez grande. On tire au hasard un point $(a_1,\dots,a_{n-1})\in S^{n-1}$, on factorise ensuite $F(a_1Y,\dots,a_{n-1}Y,Z)$, puis on utilise la remontée de Newton–Hensel en précision $\eta+1$ pour reconstruire les facteurs de F. Le lemme précédent nous assure que l'algorithme aboutit à un résultat correct avec une probabilité $\geq 1-1/C$. Si un facteur remonté en précision $\eta+1$ divise F, alors c'est un facteur irréductible de F. Si tel n'est pas le cas, alors il faut prendre un autre point $(a_1,\dots,a_{n-1})\in S^{n-1}$ au hasard et recommencer. Ainsi il est donc aisé d'assurer que l'algorithme renvoie toujours un résultat correct. Le nombre de tentatives est en moyenne borné par une constante.

Cette approche probabiliste ne peut pas être appliquée lorsque le cardinal de $\mathbb{K}$ est trop petit. Nous avons déjà rencontré cette problématique au début de ce chapitre. Nous décrivons une solution dans la section suivante.

22.5 Cas particulier des corps finis

Si $\mathbb{K}$ est un corps fini $\mathbb{F}_q$ de cardinal trop petit pour pouvoir appliquer les résultats vus dans ce chapitre, alors nous pouvons procéder comme suit. Tout d'abord nous construisons une extension algébrique $\mathbb{F}_{q^l}$ de $\mathbb{F}_q$ de degré l sous la forme $\mathbb{F}_q[Z]/(\chi(Z))$, avec χ irréductible de degré l. Notons α une racine de χ dans $\mathbb{F}_{q^l}$. En prenant $l\in O(\log_q(\eta d))$ suffisamment grand pour pouvoir factoriser F dans $\mathbb{F}_{q^l}$ avec les algorithmes de ce chapitre, nous pouvons obtenir les facteurs irréductibles $\mathcal{F}_1,\dots,\mathcal{F}_t$ de F dans $\mathbb{F}_{q^l}$. En prenant les préimages des coefficients des $\mathcal{F}_i$ dans $\mathbb{F}_q[Z]$, nous construisons des polynômes $F_i(X_1,\dots,X_n,Z)$ de degrés au plus $l-1$ en Z, tels que $\mathcal{F}_i(X_1,\dots,X_n)=F_i(X_1,\dots,X_n,\alpha)$.

Pour chaque $i\in\{1,\dots,t\}$, nous pouvons calculer

$$G_i=\operatorname{pgcd}(F,\operatorname{Res}_Z(F_i(X_1,\dots,X_n,Z),\chi(Z))),$$

qui s'avère être un facteur irréductible de F. En effet, considérons un facteur irréductible H de G_i. Par la formule de Poisson (Théorème 6.5), le polynôme

$$\operatorname{Res}_Z(F_i(X_1,\dots,X_n,Z),\chi(Z))$$

est proportionel à

$$\prod_{\chi(\beta)=0}F_i(X_1,\dots,X_n,\beta).$$

Le facteur H est donc divisible par l'un des $F_i(X_1,\dots,X_n,\beta)$ et donc par tous. Ceci nous donne $H=G_i$. Par ailleurs, des arguments similaires montrent que chaque facteur

irréductible de F apparaît au moins une fois comme l'un des G_i, et au plus l fois. Les redondances sont éliminées simplement, en triant la liste des G_i.

Pour construire le polynôme irréductible χ, on peut factoriser des polynômes de degré l pris au hasard, jusqu'à en trouver un irréductible. Nous ne justifierons pas le fait que l'on trouve rapidement un polynôme irréductible en toute généralité. Néanmoins on peut ici accepter de prendre l premier, de sorte que la Proposition 19.8 permet d'obtenir que le nombre de polynômes unitaires irréductibles de degré l est $(q^l - q)/l$.

Lorsque $n = 2$, on admettra que chaque G_i se calcule par évaluation-interpolation rapide en $\tilde{O}(\eta d)$ opérations dans $\mathbb{F}_{q^l}$. Au final le surcoût total pour résoudre les problèmes avec les corps finis trop petits reste donc borné par un facteur acceptable $\log^{O(1)}(\eta d)$.

Notes

La factorisation des polynômes à plusieurs variables est un sujet d'étude important en calcul formel depuis les années soixante-dix. L'utilisation de la remontée de Newton–Hensel à plusieurs variables a débuté avec les articles de Musser, Wang et Rothschild [Mus75 ; Wan78 ; WR75]. Des améliorations ont ensuite été trouvées par von zur Gathen et Kaltofen [Gat84 ; GK85a ; Kal85d]. Une implémentation très efficace de ces idées a été réalisée par Bernardin et Monagan [BM97].

Dans le chapitre précédent, nous avons étudié le coût binaire de la factorisation sans carré dans $\mathbb{Q}[X]$. Cette approche peut être aisément adaptée au cas de deux variables, par évaluation/interpolation, dès lors que le corps de base contient suffisamment d'éléments : on obtient un coût $\tilde{O}(\eta d^2)$, où η et d représentent les degrés partiels. De plus, en acceptant des choix aléatoires au moment de l'exécution, on peut atteindre un coût quasi-linéaire en moyenne [Lec08].

Pendant longtemps, l'étape de recombinaison a été effectuée par la méthode de recherche exhaustive, qui a un coût exponentiel dans le pire cas. Néanmoins, si $\mathbb{K}$ est un corps fini, en moyenne sur l'ensemble des polynômes, son coût est essentiellement quadratique [GL02], ce qui justifie le très bon comportement pratique de cette approche.

Les premiers algorithmes de factorisation en temps polynomiaux sont dus à Kaltofen au début des années quatre-vingt [Kal82a ; Kal82b ; Kal82c ; Kal85a ; Kal85b ; Kal85c ; Kal85d]. Plusieurs autres auteurs ont contribué à ce problème pour couvrir les corps de coefficient usuels : Chistov, von zur Gathen, Grigoriev et Lenstra. Ces algorithmes calculent un facteur rationnel à partir d'un seul facteur local, sans utiliser la totalité de la décomposition locale. Malheureusement les exposants des bornes de complexité sont grands et les seuils d'efficacité pratique sont très élevés. Les détails historiques sont présentés dans des articles de synthèse de von zur Gathen et Kaltofen [Gat06 ; Kal03 ; Kal90 ; Kal92].

La première réduction de la factorisation de deux à une variable en temps quasi-quadratique, pour la caractéristique nulle ou suffisamment grande, est due à Shuhong Gao [Gao03] par un calcul direct du premier groupe de cohomologie de De Rham du complémentaire de l'hypersurface définie par $F = 0$, dans la continuité des travaux de Ruppert [Rup86 ; Rup99] sur les tests d'irréductibilité.

Dans le présent chapitre nous nous sommes concentrés sur les méthodes utilisant la remontée de Newton–Hensel. L'Algorithme 22.2 est dû à Belabas, van Hoeij, Klüners et Steel [Bel+09]. La première borne de précision linéaire en le degré total de F a été prouvée par Bostan, Lecerf, Salvy, Schost et Wiebelt [Bos+04b], puis a été abaissée par Lecerf [Lec06]. L'algorithme de Lecerf présenté ici est issu de l'article de 2010 [Lec10], qui traite aussi les autres cas de petite caractéristique efficacement, avec les meilleures bornes de complexités connues à l'heure actuelle.

Le Théorème 22.9 est une forme quantitative d'un résultat plus général classique présenté par exemple dans le livre de Shafarevich [Sha94, Chapter II, Section 6.1]. En fait, ce cas particulier semble être dû à Hilbert [Hil92, p. 117], comme souligné par Kaltofen [Kal95a]. Pour cette raison, il est fréquent de trouver les terminologies « théorème de Hilbert » ou « théorème de Bertini » dans la littérature.

L'utilisation du théorème de Bertini en calcul formel a été mise en avant pour la première fois dans les travaux de Heintz et Sieveking d'une part [HS81], et de Kaltofen d'autre part [Kal82c]. Son utilisation s'est rapidement imposée dans de nombreux algorithmes et études de complexité [Gat85 ; GK85b ; Kal85a ; Kal85b ; Kal85c ; Kal85d]. Plusieurs auteurs ont contribué à obtenir des bornes de probabilité de plus en plus fines [Baj+93 ; Chè04 ; Gao03 ; Kal95a]. Les meilleurs résultats connus actuellement sont dûs à Lecerf [Lec07 ; Lec13].

Pour davantage de détails historiques concernant la factorisation des polynômes à plusieurs variables nous renvoyons le lecteur aux livres de Zippel [Zip93], Schinzel [Sch00], von zur Gathen et Gerhard [GG03], Mullen et Panario [MP13], mais aussi aux notes de cours de Lecerf [Lec13].

Bibliographie

Baj+93 Bajaj, Chanderjit, John Canny, Thomas Garrity et Joe Warren (1993). « Factoring rational polynomials over the complex numbers ». In : *SIAM Journal on Computing*, vol. 22, n°2, p. 318–331.

Bel+09 Belabas, Karim, Mark van Hoeij, Jürgen Klüners et Allan Steel (2009). « Factoring polynomials over global fields ». In : *Journal de Théorie des Nombres de Bordeaux*, vol. 21, n°1, p. 15–39.

BM97 Bernardin, Laurent et Michael B. Monagan (1997). « Efficient multivariate factorization over finite fields ». In : *Applied algebra, algebraic algorithms and error-correcting codes (Toulouse, 1997)*. Vol. 1255. Lecture Notes in Computer Science. Springer-Verlag, p. 15–28.

Bos+04b Bostan, Alin, Grégoire Lecerf, Bruno Salvy, Éric Schost et Bernd Wiebelt (2004). « Complexity issues in bivariate polynomial factorization ». In : *ISSAC'04 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 42–49.

Chè04 Chèze, Guillaume (2004). « Des méthodes symboliques-numériques et exactes pour la factorisation absolue des polynômes en deux variables ». Thèse de doctorat. Université de Nice-Sophia Antipolis (France).

Gao03 Gao, Shuhong (2003). « Factoring multivariate polynomials via partial differential equations ». In : *Mathematics of Computation*, vol. 72, n°242, p. 801–822.

Gat06 Gathen, Joachim von zur (2006). « Who was who in polynomial factorization ». In : *ISSAC'06 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 1–2.

Gat84 — (1984). « Hensel and Newton methods in valuation rings ». In : *Mathematics of Computation*, vol. 42, n°166, p. 637–661.

Gat85 Gathen, J. von zur (1985). « Irreducibility of multivariate polynomials ». In : *Journal of Computer and System Sciences*, vol. 31, n°2. Special issue : Twenty-fourth annual symposium on the foundations of computer science (Tucson, Ariz., 1983), p. 225–264.

GG03 Gathen, Joachim von zur et Jürgen Gerhard (2003). *Modern computer algebra*. 2e éd. Cambridge University Press.

GK85a Gathen, J. von zur et E. Kaltofen (1985). « Factorization of multivariate polynomials over finite fields ». In : *Mathematics of Computation*, vol. 45, n°171, p. 251–261.

GK85b Gathen, Joachim von zur et Erich Kaltofen (1985). « Factoring sparse multivariate polynomials ». In : *Journal of Computer and System Sciences*, vol. 31, n°2. Special issue : Twenty-fourth annual symposium on the foundations of computer science (Tucson, Ariz., 1983), p. 265–287.

GL02 Gao, Shuhong et Alan G. B. Lauder (2002). « Hensel lifting and bivariate polynomial factorisation over finite fields ». In : *Mathematics of Computation*, vol. 71, n°240, p. 1663–1676.

Hil92 Hilbert, David (1892). « Über die Irreducibilität ganzer rationaler Functionen mit ganzzahligen Coefficienten ». In : *Journal für die reine und angewandte Mathematik*, vol. 110.

HS81 Heintz, Joos et Malte Sieveking (1981). « Absolute primality of polynomials is decidable in random polynomial time in the number of variables ». In : *Automata, languages and programming (Akko, 1981)*. Vol. 115. Lecture Notes in Computer Science. Springer-Verlag, p. 16–28.

Kal03 Kaltofen, Erich (2003). « Polynomial factorization : a success story ». In : *ISSAC'03 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 3–4.

Kal82a Kaltofen, E. (1982). « A polynomial-time reduction from bivariate to univariate integral polynomial factorization ». In : *23rd Annual Symposium on Foundations of Computer Science (SFCS '08)*. IEEE, p. 57–64.

Kal82b — (1982). « Polynomial factorization ». In : *Computer algebra. Symbolic and Algebraic Computation*. Éd. par B. Buchberger, G. E. Collins et R. Loos. Springer-Verlag, p. 95–113.

Kal82c Kaltofen, Erich (1982). « A polynomial reduction from multivariate to bivariate integral polynomial factorization ». In : *STOC'82 : ACM Symposium on Theory of Computing*. ACM Press, p. 261–266.

Kal85a Kaltofen, E. (1985). « Sparse Hensel lifting ». In : *Proceedings of EUROCAL '85, Vol. 2 (Linz, 1985)*. Vol. 204. Lecture Notes in Computer Science. Springer-Verlag, p. 4–17.

Kal85b Kaltofen, Erich (1985). « Effective Hilbert irreducibility ». In : *Information and Control*, vol. 66, n°3, p. 123–137.

Kal85c — (1985). « Fast parallel absolute irreducibility testing ». In : *Journal of Symbolic Computation*, vol. 1, n°1, p. 57–67.

Kal85d — (1985). « Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization ». In : *SIAM Journal on Computing*, vol. 14, n°2, p. 469–489.

Kal90 — (1990). « Polynomial factorization 1982–1986 ». In : *Computers in mathematics (Stanford, CA, 1986)*. Vol. 125. Lecture Notes in Pure and Applied Mathematics. Dekker, p. 285–309.

Kal92 — (1992). « Polynomial factorization 1987–1991 ». In : *LATIN '92 (São Paulo, 1992)*. Vol. 583. Lecture Notes in Computer Science. Springer-Verlag, p. 294–313.

Kal95a Kaltofen, E. (1995). « Effective Noether irreducibility forms and applications ». In : *Journal of Computer and System Sciences*, vol. 50, n°2, p. 274–295.

Lec06 Lecerf, Grégoire (2006). « Sharp precision in Hensel lifting for bivariate polynomial factorization ». In : *Mathematics of Computation*, vol. 75, p. 921–933.

Lec07 — (2007). « Improved dense multivariate polynomial factorization algorithms ». In : *Journal of Symbolic Computation*, vol. 42, n°4, p. 477–494.

Lec08 — (2008). « Fast separable factorization and applications ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 19, n°2, p. 135–160.

Lec10 — (2010). « New recombination algorithms for bivariate polynomial factorization based on Hensel lifting ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 21, n°2, p. 151–176.

Lec13 — (2013). « Factorisation des polynômes à plusieurs variables (Cours no. II) ». In : *Journées nationales de calcul formel*. Éd. par G. Chèze, P. Boito, C. Pernet et M. Safey el Din. Vol. 3. Les cours du CIRM n°1. Cedram, p. 1–85. url : http://ccirm.cedram.org/ccirm-bin/fitem?id=CCIRM_2013__3_1_A2_0.

MP13 Mullen, Gary L. et Daniel Panario (2013). *Handbook of finite fields*. Discrete Mathematics and Its Applications. Chapman et Hall/CRC.

Mus75 Musser, David R. (1975). « Multivariate polynomial factorization ». In : *Journal of the ACM*, vol. 22, p. 291–308.

Rup86 Ruppert, Wolfgang M. (1986). « Reduzibilität ebener Kurven ». In : *Journal für die reine und angewandte Mathematik*, vol. 369, p. 167–191.

Rup99 — (1999). « Reducibility of polynomials $f(x,y)$ modulo p ». In : *Journal of Number Theory*, vol. 77, n°1, p. 62–70.

Sch00 Schinzel, Andrzej (2000). *Polynomials with special regard to reducibility*. Vol. 77. Encyclopedia of Mathematics and its Applications. Cambridge University Press.

Sha94 Shafarevich, I. R. (1994). *Basic algebraic geometry. 1 Varieties in projective space*. 2e éd. Springer-Verlag.

Wan78 Wang, Paul S. (1978). « An improved multivariate polynomial factoring algorithm ». In : *Mathematics of Compuptation*, vol. 32, n°144, p. 1215–1231.

WR75 Wang, Paul S. et Linda Preiss Rothschild (1975). « Factoring multivariate polynomials over the integers ». In : *Mathematics of Computation*, vol. 29, p. 935–950.

Zip93 Zippel, Richard (1993). *Effective polynomial computation*. Kluwer Academic Publishers.

V Systèmes polynomiaux

23. Bases standard

Résumé

Les bases standard de la théorie d'Hironaka pour l'étude de systèmes de fonctions analytiques s'avèrent être un outil central pour rendre effectives les manipulations de systèmes d'équations polynomiales. Elle sont alors le plus souvent connues sous la dénomination de *bases de Gröbner*.

Exemple 23.1 Le théorème d'Apollonius (Figure 23.1) affirme que les milieux des côtés d'un triangle rectangle, le sommet de l'angle droit et le pied de la hauteur relative à l'hypoténuse sont sur un même cercle. C'est une version simplifiée du théorème du cercle des 9 points d'un triangle. Les méthodes présentées dans ce chapitre permettent de prouver automatiquement ce type de résultat.

La première étape consiste à coder ce problème géométrique par des équations polynomiales. Soit donc OAB un triangle rectangle de sommet O. Nous choisissons un système de coordonnées tel que l'origine soit en O, et les deux autres sommets soient $(2a, 0)$ et $(0, 2b)$. Le cercle (C) passant par le sommet et les milieux $(a, 0)$ et $(0, b)$ des côtés a pour équation $\mathrm{C}(\mathrm{X}, \mathrm{Y}) = 0$, avec

$$\mathrm{C}(\mathrm{X}, \mathrm{Y}) = \mathrm{X}^2 + \mathrm{Y}^2 - a\mathrm{X} - b\mathrm{Y}.$$

Vérifier que le milieu (a, b) de l'hypoténuse appartient à ce cercle revient tester si $\mathrm{C}(a, b)$ est nul, ce qui est immédiat. Le pied de la hauteur, $\mathrm{H} = (x, y)$, est défini par des relations linéaires exprimant son appartenance à AB et l'orthogonalité entre OH et AB. Il est donc solution du système linéaire $\mathrm{H}_1(\mathrm{X}, \mathrm{Y}) = \mathrm{H}_2(\mathrm{X}, \mathrm{Y}) = 0$, avec

$$\mathrm{H}_1(\mathrm{X}, \mathrm{Y}) = a\mathrm{Y} + b\mathrm{X} - 2ab \quad \text{et} \quad \mathrm{H}_2(\mathrm{X}, \mathrm{Y}) = b\mathrm{Y} - a\mathrm{X}.$$

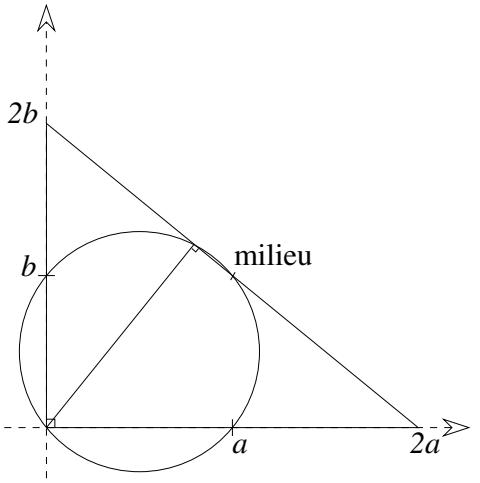


FIGURE 23.1 – Le théorème d'Apollonius.

Le théorème est donc démontré si nous prouvons que les hypothèses $H_1(x,y) = H_2(x,y) = 0$ impliquent $C(x,y) = 0$, c'est-à-dire que H appartient au cercle (C). En adjoignant les paramètres a et b au corps de base, nous verrons à l'Exemple 23.11 comment les outils présentés dans cette partie permettent d'obtenir automatiquement l'existence de deux polynômes $p(X,Y)$ et $q(X,Y)$ tels que

$$C(X,Y) = p(X,Y)H_1(X,Y) + q(X,Y)H_2(X,Y),$$

ce qui permet de conclure : si (x,y) annule H_1 et H_2, alors il annule C.

Exemple 23.2 Dans un autre ordre d'idées, le cercle unité est donné par la paramétrisation classique

$$x = \frac{1-t^2}{1+t^2} \quad \text{et} \quad y = \frac{2t}{1+t^2}.$$

Les outils présentés dans cette partie permettent de retrouver l'équation du cercle par recombinaison algorithmique des polynômes

$$(1+T^2)X - (1-T^2) \quad \text{et} \quad (1+T^2)Y - 2T$$

avec des coefficients polynomiaux en X, Y et T, ce qui correspond à *éliminer* T entre ces deux polynômes.

Plus généralement, les systèmes d'équations polynomiales sont un outil de modélisation. Par exemple, en robotique, la position de la main d'un robot en fonction de la longueur de ses bras est caractérisée par des équations polynomiales. D'autres domaines d'application sont le traitement du signal, la vision par ordinateur, la cryptologie ainsi que la chimie ou la biologie. Pour ces deux derniers domaines, on imagine

un système dont l'évolution est régie par un ensemble d'équations différentielles à coefficients polynomiaux. Les états d'équilibres, décrits par l'annulation des dérivées, sont donc donnés par des systèmes d'équations polynomiales.

Des cas particuliers importants ont été abordés dans des chapitres précédents. Ainsi, dans le cas du degré 1 (des équations affines), tester si un polynôme est combinaison linéaire à coefficients constants d'autres polynômes se réduit à la résolution d'un système linéaire (voir Chapitre 8). Dans le cas où le degré est arbitraire mais tous les polynômes sont en une seule et même variable, l'application successive de l'algorithme d'Euclide (Chapitre 6) permet de calculer de proche en proche leur plus grand commun diviseur, et ce même problème se réduit ensuite à une division euclidienne.

Ce chapitre présente une généralisation à plusieurs variables, à la fois de l'algorithme d'Euclide et de l'algorithme de Gauss, sous le vocable de « bases standard », plus souvent appelées « bases de Gröbner » dans le cas des algèbres de polynômes.

23.1 Lien entre algèbre et géométrie

La manipulation des systèmes d'équations polynomiales prend tout son sens quand elle donne des informations sur l'ensemble des solutions, à savoir le lieu des zéros communs aux équations polynomiales. Ainsi, un des grands intérêts de la combinatoire des anneaux polynomiaux réside dans la description de son substrat géométrique.

Le cas affine

Soit n le nombre de variables et $\mathrm{R} = k[\mathrm{X}_1, \ldots, \mathrm{X}_n]$ l'anneau de polynômes sur un corps effectif k (le plus souvent $\mathbb{Q}$). La notation $\mathbb{A}_k^n$, ou $\mathbb{A}^n$, représente l'*espace affine* à n dimensions sur k, c'est-à-dire l'ensemble k^n. Soient aussi $f_1, \ldots, f_s \in \mathrm{R}$ des polynômes.

Définition 23.1 La *variété affine* $\mathbf{V}_k(\mathrm{J})$ *de* $\mathbb{A}_k^n$ *associée à un ensemble de polynômes* $\mathrm{J} \subseteq \mathrm{R}$ est l'ensemble des racines dans k communes aux polynômes de J :

$$\mathbf{V}_k(\mathrm{J}) := \Big\{(x_1, \ldots, x_n) \in \mathbb{A}^n \mid \forall f \in \mathrm{J}, f(x_1, \ldots, x_n) = 0\Big\}.$$

On appelle « variété algébrique affine » tout sous-ensemble de $\mathbb{A}^n$ obtenu de cette manière. S'il n'y a pas d'ambiguïté sur le corps k, nous noterons $\mathbf{V}(\mathrm{J}) := \mathbf{V}_k(\mathrm{J})$. De plus, si J est un ensemble fini $\{f_1, \ldots, f_k\}$, nous noterons souvent $\mathbf{V}(f_1, \ldots, f_k)$ pour $\mathbf{V}(\mathrm{J})$.

Dans cet ouvrage, les variétés que nous considérons sont majoritairement algébriques. Donc, lorsque le contexte est clair, nous omettons le qualificatif « algébrique ».

Exemple 23.3 La variété réelle $\mathbf{V}_{\mathbb{R}}(\mathrm{X}^2 + \mathrm{Y}^2 + 1)$ est vide. Cependant la variété complexe $\mathbf{V}_{\mathbb{C}}(\mathrm{X}^2 + \mathrm{Y}^2 + 1)$ est non vide.

Définition 23.2 Un *idéal* I d'un anneau R est un sous-groupe additif de R stable par multiplication par les éléments de R. C'est-à-dire que $I \neq \varnothing$ et $\forall f, g \in I$, $f - g \in I$ et $\forall f \in I, \forall h \in R$, $hf \in I$.

L'idéal engendré par une partie $\{f_1, \ldots, f_s\}$ de R est l'ensemble $\sum_{i=1}^{s} Rf_i$. Il est noté $(f_1, \ldots, f_s)$.

On a $\mathbf{V}\big(\{f_1, \ldots, f_s\}\big) = \mathbf{V}(I)$ avec $I = (f_1, \ldots, f_s)$, c'est-à-dire que si deux ensembles de polynômes engendrent le même idéal alors ils définissent la même variété. Ainsi l'idéal $(f_1, \ldots, f_s)$ est l'objet naturel attaché au système d'équations $f_1 = 0, \ldots, f_s = 0$. Le résultat suivant découle de ce que l'opération $\mathbf{V}$ met en place une sorte de dualité.

Lemme 23.1 Pour deux ensembles J_1 et J_2 de polynômes, si $J_1 \subseteq J_2$, alors $\mathbf{V}(J_1) \supseteq \mathbf{V}(J_2)$.

De même qu'à tout idéal correspond une variété, à toute variété correspond son idéal annulateur :

Définition 23.3 L'*idéal annulateur* $\mathbf{I}_k(E)$ associé à un sous-ensemble E de $\mathbb{A}_k^n$ est l'ensemble des polynômes de R s'annulant identiquement sur E, c'est-à-dire

$$\mathbf{I}_k(E) = \{f \in R \mid \forall x \in E, f(x) = 0\}.$$

Si il n'y a pas d'ambiguïté sur le corps k, nous notons $\mathbf{I}(E) := \mathbf{I}_k(E)$.

On vérifie facilement que $\mathbf{I}(E)$ est un idéal.

Exemple 23.4 L'hyperbole d'équation $XY = 1$ est la variété $\mathbf{V}(XY - 1)$. De même, le cercle d'équation $X^2 + Y^2 = 1$ est une variété.

La dualité évoquée plus haut se prolonge avec les résultats suivants.

Lemme 23.2 Pour deux sous-ensembles E_1 et E_2 de $\mathbb{A}^n$, si $E_1 \subseteq E_2$, alors $\mathbf{I}(E_1) \supseteq \mathbf{I}(E_2)$. Si J est un ensemble de polynômes et E une partie de $\mathbb{A}^n$, alors $J \subseteq \mathbf{I}\big(\mathbf{V}(J)\big)$ et $E \subseteq \mathbf{V}\big(\mathbf{I}(E)\big)$.

Exemple 23.5 Si $R = k[X]$ et $I = (X)$, alors $\mathbf{V}(I) = \{0\}$ et $\mathbf{I}(\mathbf{V}(I)) = I$. Par contre, si $J = (X^2)$ alors $\mathbf{V}(J) = \mathbf{V}((X))$ et $\mathbf{I}\big(\mathbf{V}(J)\big) = (X)$ n'est pas égal à J.

Exemple 23.6 Si $R = \mathbb{Q}[X, Y]$ et $E = (k \setminus \{0\}) \times \{0\}$, alors $\mathbf{I}(E) = (Y)$ et $\mathbf{V}(\mathbf{I}(E)) = k \times \{0\}$ n'est pas égal à E.

Les variétés affines sont des objets géométriques et il est fructueux de les manipuler en suivant une intuition géométrique. La démonstration algébrique traduit souvent

cette intuition.

Exercice 23.1 Montrer que l'union de deux variétés affines d'un même espace affine est une variété affine. Montrer que $V \subseteq \mathbb{A}^n$ est une variété affine si, et seulement si, on a $\mathbf{V}(\mathbf{I}(V)) = V$. ∎

Le cas projectif

Le passage à la géométrie projective est motivé par la simplification des énoncés et de certains algorithmes dans ce cadre. Dans le cadre projectif l'anneau des polynômes comporte une variable de plus notée X_0, mais nous continuons de noter R l'anneau ambiant de polynômes $k[X_0,\dots,X_n]$.

Définition 23.4 Nous notons $\mathbb{P}^n_k$, ou $\mathbb{P}^n$, l'*espace projectif* de dimension n sur le corps k, c'est-à-dire l'ensemble défini comme le quotient de $k^{n+1}\setminus\{0\}$ par la relation d'équivalence $(x_0,\dots,x_n) \sim (y_0,\dots,y_n)$ si et seulement si $\exists \lambda \in k,\ \forall i,\ x_i = \lambda y_i$.

On note $(x_0 : \cdots : x_n)$ la classe d'équivalence de $(x_0,\dots,x_n)$, que l'on appelle aussi un point (projectif).

Soit $U_i = \left\{(a_0 : \cdots : a_n) \mid a_i \neq 0\right\}$ pour $i \in \{0,\dots,n\}$. Les U_i sont des ouverts de $\mathbb{P}^n$ qui sont en bijection avec l'espace affine $\mathbb{A}^n$ via

$$\begin{array}{ccc} \mathbb{A}^n & \to & U_i \\ (x_1,\dots,x_n) & \mapsto & (x_0 : \cdots : x_{i-1} : 1 : x_{i+1} : \cdots : x_n) \end{array}.$$

Ces U_i recouvrent l'espace projectif. L'ensemble U_i est appelé la *i-ième partie affine* de $\mathbb{P}^n$.

Définition 23.5 On appelle *points à l'infini* de $\mathbb{P}^n$ les points de $\mathbb{P}^n \setminus U_0$. Ils correspondent aux directions de droites dans $\mathbb{A}^n$.

Le formalisme ne doit pas faire oublier que l'espace projectif est un objet naturel : c'est l'espace affine auquel on a ajouté des points à l'infini. Ces points à l'infini sont les points $(0 : x_1 : \cdots : x_n)$, alors que les points affines sont de la forme $(1 : x_1 : \cdots : x_n)$. On a donc

$$\mathbb{P}^n = \left\{(1 : x_1 : \cdots : x_n) \;\middle|\; x \in \mathbb{A}^n\right\} \cup \left\{(0 : x_1 : \cdots : x_n) \;\middle|\; x \in \mathbb{A}^n \setminus \{0\}\right\}.$$

Dans un espace projectif, construire des équations à partir de polynômes est plus délicat que dans le cas affine. Soit $f \in R$ un polynôme homogène, c'est-à-dire tel que tous ses monômes soient de même degré total. Alors $f(x_0 : \cdots : x_n) = 0$ a un sens car f s'annule soit sur tout représentant de $(x_0 : \cdots : x_n)$, soit sur aucun d'entre eux. Ainsi à tout polynôme homogène f de R on peut associer une équation $f = 0$ sur $\mathbb{P}^n$. Il faut bien noter que R est un anneau de polynômes à $n+1$ variables, alors que n est la dimension de l'espace projectif.

Définition 23.6 La *variété projective* $\mathbf{V}(\mathrm{J})$ *associée à l'ensemble de polynômes homogènes* $\mathrm{J} \subseteq \mathrm{R}$ est l'ensemble des solutions communes aux équations homogènes, c'est-à-dire $\left\{(x_0 : \cdots : x_n) \in \mathbb{P}^n_k \mid \forall j \in \mathrm{J},\ j(x_0 : \cdots : x_n) = 0\right\}$.

Définition 23.7 Un idéal de R est dit *homogène* s'il est engendré par des polynômes homogènes.

Proposition 23.3 Un idéal I de R est homogène si, et seulement si, l'espace vectoriel I est la somme directe $\bigoplus_{d\geq 0} \mathrm{I}_d$, où I_d est l'espace vectoriel des polynômes de I homogènes de degré d.

Démonstration. Supposons que I soit engendré par des polynômes homogènes $f_1, \ldots, f_s$ et soit h un polynôme de I. Il existe alors des polynômes $g_1, \ldots, g_s$ tels que $h = g_1 f_1 + \cdots + g_s f_s$. En notant $g_{i,d}$ la composante homogène de degré d de g_i, l'égalité précédente se récrit

$$h = \sum_{d\geq 0} g_{1,d} f_1 + \cdots + \sum_{d\geq 0} g_{s,d} f_s,$$

et la composante homogène h_d de h s'obtient alors comme

$$h_d = g_{1,d-\deg f_1} f_1 + \cdots + g_{s,d-\deg f_s} f_s.$$

Autrement dit, les composantes homogènes de h sont aussi dans I, d'où $\mathrm{I} = \bigoplus_{d\geq 0} \mathrm{I}_d$.

Réciproquement, si $\mathrm{I} = \bigoplus_{d\geq 0} \mathrm{I}_d$, et si I est engendré par les polynômes $f_1, \ldots, f_s$ alors les composantes homogènes des f_i sont dans I et par conséquent l'engendrent. ■

L'homogénéisation des polynômes et des idéaux permet de prolonger naturellement une variété affine dans l'espace projectif associé.

Définition 23.8 L'*homogénéisé* $f^\sharp \in \mathrm{R}$ d'un polynôme $f \in k[\mathrm{X}_1, \ldots, \mathrm{X}_n]$ est le polynôme $\mathrm{X}_0^{\deg(f)} f(\mathrm{X}_1/\mathrm{X}_0, \ldots, \mathrm{X}_n/\mathrm{X}_0)$. L'*idéal homogénéisé* $\mathrm{I}^\sharp$ d'un idéal I de $k[\mathrm{X}_1, \ldots, \mathrm{X}_n]$ est l'idéal de R engendré par $\{f^\sharp \mid f \in \mathrm{I}\}$.

Exemple 23.7 Le cercle $\mathbf{V}_{\mathbb{C}}(\mathrm{X}_1^2 + \mathrm{X}_2^2 - 1)$ a pour idéal annulateur $\mathrm{I} = (\mathrm{X}_1^2 + \mathrm{X}_2^2 - 1)$. Son homogénéisé est $\mathrm{I}^\sharp = (\mathrm{X}_1^2 + \mathrm{X}_2^2 - \mathrm{X}_0^2)$, qui annule la variété projective

$$\mathbf{V}_{\mathbb{C}}(\mathrm{I}^\sharp) = \{(1 : x : y) \mid (x, y) \in \mathbf{V}(\mathrm{I})\} \cup \{(0 : 1 : i), (0 : 1 : -i)\},$$

c'est-à-dire que le prolongement du cercle dans l'espace projectif $\mathbb{P}^2_{\mathbb{C}}$ contient deux points à l'infini, classiquement appelés les points cycliques.

L'opération de *déshomogénéisation* associe au polynôme $g \in k[X_0,\dots,X_n]$ le polynôme $g^\flat = g(1,X_1,\dots,X_n) \in k[X_1,\dots,X_n]$. On vérifie facilement que pour $g \in k[X_1,\dots,X_n]$ on a $(g^\sharp)^\flat = g$. Cependant, la composition inverse n'est pas l'identité : $(X_0^\flat)^\sharp = 1$. La proposition suivante fait le lien entre $(g^\flat)^\sharp$ et g.

Proposition 23.4 Soit $g \in k[X_0,\dots,X_n]$ un polynôme homogène, alors il existe $e \in \mathbb{N}$ tel que $X_0^e (g^\flat)^\sharp = g$.

Démonstration. Soit $d = \deg g$. Voyons g comme un polynôme en X_0 et notons e sa valuation en X_0 : g s'écrit $g = \sum_{i=0}^{d-e} g_{d-e-i} X_0^{e+i}$ avec $g_i \in k[X_1,\dots,X_n]$ de degré i. Ainsi $g^\flat = \sum_{i=0}^{d-e} g_{d-e-i}$ et $d = \deg g^\flat + e$. Finalement $X_0^e (g^\flat)^\sharp = X_0^e \sum_{i=0}^{d-e} g_{d-e-i} X_0^i = g$. ■

23.2 Ordres totaux admissibles sur le monoïde des monômes

Dans l'algorithme de division d'Euclide, une étape élémentaire consiste à tuer le terme de plus haut degré du dividende par le terme de plus haut degré du diviseur. Afin de concevoir un analogue en plusieurs variables, il faut d'abord définir une relation d'ordre qui permette de généraliser la notion de terme de plus haut degré. Ceci s'obtient assez simplement en considérant les exposants des monômes et des relations d'ordre convenables. Notons $R = k[X_1,\dots,X_n]$. On appelle *monôme* de R un produit de puissances $X = X_1^{a_1} \cdots X_n^{a_n}$ associé à un élément $a = (a_1,\dots,a_n)$ de $\mathbb{N}^n$. Selon les cas, il sera commode de considérer soit le monôme, soit le n-uplet de ses exposants. Dans les deux cas, la structure sous-jacente est la suivante.

Définition 23.9 Un *monoïde* $\mathcal{M}$ est un ensemble qui vérifie les axiomes des groupes à l'exception de l'existence de l'inverse pour un élément quelconque.

Ainsi les monômes de l'anneau R forment un monoïde multiplicatif $\mathcal{M}_n$ isomorphe au monoïde additif $\mathbb{N}^n$. Dans cet exemple, l'élément neutre $1 \in \mathcal{M}_n$ correspond au point de coordonnées toutes nulles dans $\mathbb{N}^n$.

Définition 23.10 Un ordre total $<$ sur les monômes est dit *compatible* s'il est compatible avec la structure de monoïde, au sens où la multiplication par un monôme est croissante.

Définition 23.11 Un ordre compatible est dit être un *ordre admissible* si l'élément neutre est plus petit que tous les autres monômes.

R Si l'élément neutre est plus grand que tous les autres monômes, nous obtenons un ordre total adapté à la manipulation des séries. Le substrat géométrique devient alors local et correspond à la théorie originelle de Hironaka, mais cela nous conduirait largement au-delà de cet ouvrage.

Définition 23.12 Un ordre compatible est dit être un *bon ordre* si toute suite décroissante de monômes stationne. En résumé, un ordre $<$ sur les monômes est un bon ordre si :

1. c'est un ordre total ;
2. pour tous m, m' et m'', $m' < m''$ implique $mm' < mm''$.
3. pour toute suite décroissante de monômes $(m_i)_{i\in\mathbb{N}}$, il existe un rang N tel que pour tout $n \geq \mathrm{N}$, on ait $m_n = m_{\mathrm{N}}$.

Un bon ordre est toujours admissible car on peut construire une suite infinie strictement décroissante à partir des puissances d'un monôme $m < 1$. Le Corollaire 23.8 ci-dessous montre qu'un ordre admissible est un bon ordre.

Définition 23.13 Soit σ une permutation de $\{1,\dots,n\}$. L'*ordre lexicographique* induit par l'ordre $\mathrm{X}_{\sigma(n)} < \mathrm{X}_{\sigma(n-1)} < \cdots < \mathrm{X}_{\sigma(1)}$ sur les variables est défini par : $\mathrm{X}_1^{a_1}\cdots\mathrm{X}_n^{a_n} < \mathrm{X}_1^{b_1}\cdots\mathrm{X}_n^{b_n}$ si

— $a_{\sigma(1)} < b_{\sigma(1)}$, ou
— $a_{\sigma(1)} = b_{\sigma(1)}$ et $a_{\sigma(2)} < b_{\sigma(2)}$, ou
— $a_{\sigma(1)} = b_{\sigma(1)}$, $a_{\sigma(2)} = b_{\sigma(2)}$ et $a_{\sigma(3)} < b_{\sigma(3)}$, ou
— …, ou
— $a_{\sigma(1)} = b_{\sigma(1)}$,…, $a_{\sigma(n-1)} = b_{\sigma(n-1)}$ et $a_{\sigma(n)} < b_{\sigma(n)}$.

L'ordre lexicographique *usuel*, noté $<_{\mathrm{lex}}$, est celui induit par $\mathrm{X}_n <_{\mathrm{lex}} \mathrm{X}_{n-1} <_{\mathrm{lex}} \cdots <_{\mathrm{lex}} \mathrm{X}_1$. Il est noté `plex(x[1], ..., x[n])` en Maple (pour *pure lexicographic*). Les plus grands monômes sont donc ceux qui ont le plus de X_1, puis le plus de X_2, puis de X_3 etc. Les autres ordres lexicographiques se ramènent à l'ordre usuel en permutant les variables.

Lemme 23.5 Les ordres lexicographiques sont de bons ordres.

Démonstration. Sans perte de généralité nous pouvons considérer l'ordre usuel. La compatibilité de l'ordre lexicographique est facile à établir. Montrons le stationnement de toute suite décroissante par récurrence sur le nombre n de variables. Pour $n = 1$ c'est une propriété fondamentale de $\mathbb{N}$. Soit une suite $(a_j)_{j\in\mathbb{N}}$ décroissante pour $<_{\mathrm{lex}}$ dans $\mathbb{N}^n$. Écrivons chaque a_j sous la forme (α_j, b_j) pour $\alpha_j \in \mathbb{N}$ et $b_j \in \mathbb{N}^{n-1}$. La suite des α_j est alors décroissante et stationne, par le cas $n = 1$. La suite des b_j est donc décroissante à partir d'un certain rang, et donc stationne. En conclusion, la suite $(a_j)_{j\in\mathbb{N}}$ stationne elle aussi. ∎

Définition 23.14 L'*ordre gradué* induit par un ordre compatible $<$ est l'ordre noté $a <_d b$ pour $a = (a_1,\dots,a_n)$ et $b = (b_1,\dots,b_n)$ si $\sum a_i < \sum b_i$ ou $\left(\sum a_i = \sum b_i \text{ et } a < b\right)$.

Définition 23.15 L'*ordre lexicographique gradué*, noté $<_{\mathrm{grlex}}$, est l'ordre gradué induit par l'ordre lexicographique usuel.

En Maple, ce dernier ordre est noté `grlex(x[1], ..., x[n])` (comme abbréviation de la traduction anglaise « graded lexicographic order »). La propriété de bon ordre est triviale pour les ordres gradués, par finitude du nombre de monômes de même degré.

Définition 23.16 L'*ordre lexicographique renversé*, noté $<_{\mathtt{revlex}}$, est l'opposé de l'ordre lexicographique, noté $\mathtt{lex}'$, induit par $X_1 <_{\mathtt{lex}'} X_2 <_{\mathtt{lex}'} \cdots <_{\mathtt{lex}'} X_n$.

Autrement dit, $(a_1,\dots,a_n) <_{\mathtt{revlex}} (b_1,\dots,b_n)$ correspond à $(a_1,\dots,a_n) >_{\mathtt{lex}'} (b_1,\dots,b_n)$. Les plus grands monômes sont donc ceux qui ont le moins de X_n, puis le moins de X_{n-1}, puis de X_{n-2}, etc. Cet ordre n'est pas un bon ordre : la suite des mônomes X_1^i est décroissante mais ne stationne pas.

Définition 23.17 L'ordre *lexicographique renversé gradué*, noté $<_{\mathtt{tdeg}}$, est l'ordre gradué induit par l'ordre lexicographique renversé.

Cet ordre est un bon ordre, puisqu'il est gradué. Ainsi, dans l'espace des exposants, on a $a <_{\mathtt{tdeg}} b$ pour $a = (a_1,\dots,a_n)$ et $b = (b_1,\dots,b_n)$ si $\sum a_i < \sum b_i$ ou $\left(\sum a_i = \sum b_i \text{ et } a <_{\mathtt{revlex}} b\right)$. Cela consiste donc à ordonner les monômes par degré total, puis les plus grands sont ceux qui contiennent le moins de X_n puis de X_{n-1} et ainsi de suite. Cet ordre est noté `tdeg(x[1], ..., x[n])` en Maple. En anglais, il est appelé « graded reverse lexicographic order », et est souvent noté « grevlex » dans la littérature.

Exemple 23.8 Nous illustrons les différences entre ces ordres sur des exemples :

— pour les ordres $<_{\mathtt{lex}}$ et $<_{\mathtt{revlex}}$, on a

$$X_3 <_{\mathtt{lex}} X_2X_4 <_{\mathtt{lex}} X_1 \quad \text{et} \quad X_2X_4 <_{\mathtt{revlex}} X_3 <_{\mathtt{revlex}} X_1 \ ;$$

— pour les ordres $<_{\mathtt{grlex}}$ et $<_{\mathtt{tdeg}}$, on a

$$\begin{aligned} X_3^3 <_{\mathtt{grlex}} X_2X_3^2 <_{\mathtt{grlex}} X_2^2X_3 <_{\mathtt{grlex}} X_2^3 <_{\mathtt{grlex}} X_1X_3^2 \\ <_{\mathtt{grlex}} X_1X_2X_3 <_{\mathtt{grlex}} X_1X_2^2 <_{\mathtt{grlex}} X_1^2X_3 <_{\mathtt{grlex}} X_1^2X_2 <_{\mathtt{grlex}} X_1^3 \end{aligned}$$

et

$$\begin{aligned} X_3^3 <_{\mathtt{tdeg}} X_2X_3^2 <_{\mathtt{tdeg}} X_1X_3^2 <_{\mathtt{tdeg}} X_2^2X_3 <_{\mathtt{tdeg}} X_1X_2X_3 \\ <_{\mathtt{tdeg}} X_1^2X_3 <_{\mathtt{tdeg}} X_2^3 <_{\mathtt{tdeg}} X_1X_2^2 <_{\mathtt{tdeg}} X_1^2X_2 <_{\mathtt{tdeg}} X_1^3. \end{aligned}$$

Exemple 23.9 — Ordre d'élimination. Soit $i \in \{1,\dots,n\}$. Le i-ième *ordre d'élimination* $<_i$ de $\mathbb{N}^n$ est l'ordre défini par $a <_i b$ si et seulement si $\sum_{j=1}^i a_j < \sum_{j=1}^i b_j$ ou $\left(\sum_{j=1}^i a_j = \sum_{j=1}^i b_j \text{ et } a <_{\mathtt{lex}} b\right)$.

Ces ordres d'élimination sont à mi-chemin entre l'ordre lexicographique usuel

(que l'on retrouve pour $i = 1$) et l'ordre lexicographique gradué (retrouvé pour $i = n$). Ils sont utilisés pour éliminer simultanément plusieurs variables entre des équations polynomiales (voir Chapitre 24).

Exemple 23.10 Une dernière méthode pour créer des ordres compatibles à partir de l'ordre lexicographique est de prendre une matrice $M \in \mathcal{M}_{m\times n}(\mathbb{R})$ et de poser $a <_M b$ si $Ma <_{\text{lex}} Mb$. Si M est injective alors l'ordre est total. L'hypothèse supplémentaire que toutes les colonnes C de M vérifient $0 <_{\text{lex}} C$ est nécessaire pour que l'ordre devienne admissible. Réciproquement, on a le résultat suivant, que nous ne prouverons pas ici : pour tout ordre $<$ admissible sur $\mathbb{N}^n$, il existe $M \in \mathcal{M}_{m\times n}(\mathbb{R})$ telle que $a < b \iff Ma <_{\text{lex}} Mb$.

Exercice 23.2 Expliciter les matrices de l'exemple précédent pour les ordres `lex`, `grlex`, et `tdeg`. ■

23.3 Exposants privilégiés et escaliers

Les ordres sur les monômes étant définis, l'étape suivante dans la construction d'une généralisation de la division euclidienne à plusieurs variables est la définition des termes de tête des polynômes, et l'analyse de l'ensemble de ces termes pour tous les polynômes d'un idéal.

Définition 23.18 À tout polynôme

$$f = \sum_{a\in\mathbb{N}^n} f_a X^a$$

est associé son *support* $\{a \in \mathbb{N}^n \mid f_a \neq 0\}$ dans $\mathbb{N}^n$.

Définition 23.19 L'*exposant privilégié* d'un polynôme f *non nul*, noté $\exp(f)$ pour un ordre donné $<$ est le plus grand n-uplet du support de f pour l'ordre $<$. Le *monôme de tête* est $\text{mt}(f) := X^{\exp(f)}$. Le *terme de tête* de f est $\text{tt}(f) := f_{\exp(f)}X^{\exp(f)}$ (un terme est le produit d'un monôme par un coefficient, c'est-à-dire est de la forme $c_\alpha X^\alpha$). Le coefficient du terme de tête est noté $\text{ct}(f)$.

La notion d'exposant privilégié et celles qui en découlent ne sont pas définies pour le polynôme nul.

Définition 23.20 Un *idéal de monômes* est une partie de $\mathcal{M}_n$ (le monoïde des monômes) stable par multiplication externe : tout multiple d'un élément de cet idéal de monômes par un monôme quelconque appartient encore à cet idéal de monômes.

De manière analogue, une *partie stable* de $\mathbb{N}^n$ est stable par addition de qua-

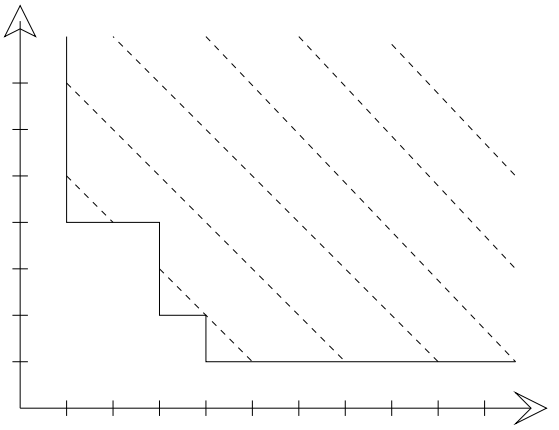


FIGURE 23.2 – Un exemple d'escalier dans $\mathbb{N}^2$.

drant : tout translaté d'un point de cette partie stable par un élément de $\mathbb{N}^n$ est encore dans cette partie stable.

On fera attention à cette terminologie : au sens de cet ouvrage, un idéal de monômes n'est pas un idéal. Certains auteurs parlent d'ailleurs de « monoïdéal » pour marquer la distinction ; d'autres auteurs réservent le vocable d'« idéal de monômes » pour les idéaux (polynomiaux) engendrés par les idéaux de monômes de cet ouvrage. Plutôt que d'idéal de monômes, on parle le plus souvent de manière imagée d'un *escalier*.

Définition 23.21 L'*escalier* $\mathrm{E}(\mathrm{I})$ associé à l'idéal I pour un ordre $<$ est la partie stable de $\mathbb{N}^n$ donnée par $\{\exp_<(f) \mid f \in \mathrm{I} \setminus \{0\}\}$.

Dans la Figure 23.2, l'escalier est donc la partie hachurée située au-dessus de la ligne qui ressemble à un escalier. Par convention, et bien que le polynôme nul ne possède pas d'exposant privilégié, $\mathrm{E}(0)$ est l'ensemble vide $\varnothing$; tandis que plus naturellement $\mathrm{E}(\mathrm{R})$ est $\mathbb{N}^n$ tout entier puisque $\mathrm{R} = (1)$.

23.4 Noethérianité et bases standard

La *noethérianité* désigne le fait que toute suite croissante est stationnaire à partir d'un certain rang.

Lemme 23.6 Les propriétés suivantes sont équivalentes :

1. Toute partie stable de $\mathbb{N}^n$ est engendrée par un nombre fini d'éléments, c'est-à-dire :

$$\mathrm{E} = \bigcup_{i=1}^{q} \left(a_i + \mathbb{N}^n\right). \tag{23.1}$$

2. Toute suite croissante de parties stables de $\mathbb{N}^n$ stationne.

Démonstration. (1) implique (2) : Soit $(\mathrm{E}_j)_{j\in\mathbb{N}}$, une suite croissante de parties stables. La réunion $\mathrm{E} = \bigcup_{j\in\mathbb{N}} \mathrm{E}_j$ est encore stable, donc finiment engendrée sous la forme (23.1). Chacun des générateurs a_i appartient à un membre E_{j_i} de la suite. Soit $\hat{\jmath}$ la valeur maximale des j_i. Comme $\mathrm{E}_{\hat{\jmath}}$ contient tous les E_j précédents, $\mathrm{E} = \bigcup_{j=1}^{q}(a_i + \mathbb{N}^n) \subseteq \sum_{j=1}^{\hat{\jmath}} \mathrm{E}_j \subseteq \mathrm{E}_{\hat{\jmath}} \subseteq \mathrm{E}$, d'où $\mathrm{E} = \mathrm{E}_{\hat{\jmath}}$. Par suite, pour tout $h \in \mathbb{N}$, $\mathrm{E} = \mathrm{E}_{\hat{\jmath}} \subseteq \mathrm{E}_{\hat{\jmath}+h} \subseteq \mathrm{E}$, et la suite stationne donc sur la valeur $\mathrm{E} = \mathrm{E}_{\hat{\jmath}}$.

(2) implique (1) : Supposons que E soit engendré par une suite dénombrable $a_1, a_2, a_3, \ldots$ On pose $\mathrm{E}_i := \bigcup_{j=1}^{i}(a_j + \mathbb{N}^n)$, pour tout $i \geq 1$. La suite des E_i est croissante et donc stationne à partir d'un certain entier $\hat{\imath}$. On en déduit $\mathrm{E} = \bigcup_{j\geq 1} \mathrm{E}_j = \bigcup_{j=1}^{\hat{\imath}} \mathrm{E}_j$. ■

Proposition 23.7 — Lemme de Dickson. Toute partie stable E de $\mathbb{N}^n$ est finiment engendrée, c'est-à-dire qu'il existe une famille $a_1, \ldots, a_q$ telle que :

$$\mathrm{E} = \bigcup_{i=1}^{q} \big(a_i + \mathbb{N}^n\big).$$

Démonstration. La démonstration se fait par récurrence sur n. L'assertion est immédiate pour $n = 1$. Si l'assertion est vraie pour $n \geq 1$, on introduit $\pi : \mathbb{N}^{n+1} \to \mathbb{N}^n$ la projection canonique sur $\mathbb{N}^n$, identifié à $\mathbb{N}^n \times \{0\}$. Chacune des sections E_j de E par l'hyperplan de coordonnées $\mathrm{X}_{n+1} = j$ s'identifie aussi par π à une partie stable de $\mathbb{N}^n$, engendrée par une famille finie, par récurrence. Les $\pi(\mathrm{E}_j)$ forment une suite croissante de parties stables qui stationne sur une valeur E_∞ à partir d'un certain indice η, par la proposition précédente. Une famille de générateurs de E est obtenue en prenant la réunion des générateurs des E_j pour $j \leq \eta$. ■

Corollaire 23.8 Tout ordre total admissible est un *bon ordre*, c'est-à-dire que toute chaîne descendante (c'est-à-dire une suite décroissante) stationne.

Démonstration. Soit une suite $(a_j)_{j\in\mathbb{N}}$ décroissante de $\mathbb{N}^n$ pour un ordre $<$ admissible vu dans le monoïde des exposants. Soit $\mathrm{E}_i := \bigcup_{j=0}^{i}(a_j + \mathbb{N}^n)$ pour tout $i \geq 0$. La suite des E_i est croissante et donc stationne à partir d'un indice N, par le lemme de Dickson. Si $k \geq \mathrm{N}$ alors a_k appartient à l'un des $a_j + \mathbb{N}^n$ pour un indice $j \leq \mathrm{N}$ et donc $a_k \geq a_j \geq a_\mathrm{N}$. On en déduit $a_k = a_\mathrm{N}$. ■

Définition 23.22 Une *base standard* ou *base de Gröbner* d'un idéal de l'anneau de polynômes est un ensemble fini de polynômes de l'idéal dont les exposants privilégiés engendrent l'escalier de l'idéal. Elle est dite *minimale* lorsque ces exposants engendrent minimalement l'escalier.

Au niveau des polynômes, cela signifie que tout polynôme de l'idéal a son terme de tête divisible par le monôme de tête d'un des éléments de la base standard.

Le lemme de Dickson donne l'existence d'une base standard pour tout idéal I de R. Notons bien que les éléments doivent appartenir à l'idéal mais que rien pour l'instant hormis la terminologie ne prouve qu'ils l'engendrent. C'est le paragraphe suivant qui va le démontrer, grâce à une généralisation à plusieurs variables de la notion de division.

Notons aussi qu'une base standard ne sera pas une base, mais seulement un système de générateurs : elle ne fournira pas une unicité de l'écriture, au sens où les cofacteurs ne sont pas définis uniquement.

Exemple 23.11 Les algorithmes du chapitre suivant montrent que l'idéal engendré par les deux polynômes $\mathrm{H}_1(\mathrm{X},\mathrm{Y}) = a\mathrm{Y} + b\mathrm{X} - 2ab$ et $\mathrm{H}_2(\mathrm{X},\mathrm{Y}) = b\mathrm{Y} - a\mathrm{X}$ de l'Exemple 23.1 possède la même base standard pour chacun des ordres `lex`, `grlex`, et `tdeg` :

$$\{(a^2+b^2)\mathrm{X} - 2ab^2, \quad (a^2+b^2)\mathrm{Y} - 2a^2b\}. \tag{23.2}$$

Dans ce cas linéaire, la base donne ainsi la solution du système. L'escalier est alors très simple : seul le point (0,0) est strictement en dessous.

23.5 Divisions

Nous pouvons maintenant étendre la division euclidienne aux polynômes à plusieurs variables une fois choisi un ordre admissible $<$ sur les monômes. Pour ce faire, nous commençons par définir une *division élémentaire faible* (resp. *forte*) d'un polynôme dividende f par un seul diviseur g non nul. Posons $r := \mathrm{tt}(g) - g$ et associons à ce diviseur la règle de récriture $\mathrm{tt}(g) \to_g r$. Elle consiste à remplacer une occurrence éventuelle de $\mathrm{tt}(g)$ comme facteur du monôme privilégié (resp. de tout monôme) du dividende par le polynôme r.

Pour être plus explicite, soit un dividende (non nul) exprimé sous la forme $f = c_1 m_1 + \cdots + c_\ell m_\ell$ pour une suite strictement décroissante de monômes m_i et des coefficients non nuls c_i. Si le monôme de tête $\mathrm{mt}(g)$ divise m_1 pour le cas de la division faible (resp. s'il divise l'un des m_i, que l'on choisit être le plus grand possible, pour le cas de la division forte), alors la division consiste à récrire f en le polynôme

$$f - \frac{\mathrm{tt}(f)}{\mathrm{tt}(g)} g = \frac{\mathrm{tt}(f)}{\mathrm{tt}(g)} r + c_2 m_2 + \cdots + c_\ell m_\ell \tag{23.3}$$

dans le cas de la division faible (resp. en le polynôme

$$f - \frac{c_i m_i}{\mathrm{tt}(g)} g = c_1 m_1 + \cdots + c_{i-1} m_{i-1} + \frac{c_i m_i}{\mathrm{tt}(g)} r + c_{i+1} m_{i+1} + c_\ell m_\ell \tag{23.4}$$

dans le cas de la division forte). Dans toutes ces écritures, chaque quotient est en fait un terme (au sens de la Définition 23.19).

La division faible consiste à appliquer cette règle de récriture à $\mathrm{mt}(f)$ (si possible) et à itérer la récriture au nouveau polynôme tant que possible. La division forte

consiste à appliquer la règle de récriture à tout terme de f et à s'arrêter quand on ne peut plus l'appliquer à aucun terme.

L'itération du processus de division faible (resp. forte) s'arrête (car $<$ est un bon ordre par le Corollaire 23.8) sur un reste dont le monôme privilégié n'est plus divisible par $\mathrm{mt}(g)$ (resp. dont aucun monôme n'est divisible par $\mathrm{mt}(g)$).

La division faible (resp. forte) par *une famille* $g_1,\dots,g_s$ consiste à appliquer successivement l'une des règles de récriture $\mathrm{tt}(g_i) \to_{g_i} r_i,\ i \in \{1,\dots,s\}$, à un dividende f jusqu'à ce qu'on ne puisse plus l'appliquer au terme de tête $\mathrm{tt}(f)$ de f (resp. à aucun terme de f).

En rassemblant de façon adéquate les monômes $\mathrm{tt}(f)/\mathrm{tt}(g_j)$ intervenant dans les étapes (23.3) d'une division faible (resp. les monômes $c_i m_i/\mathrm{tt}(g_j)$ dans les étapes (23.4) d'une division forte), on obtient des quotients q_j, un reste final r, ainsi qu'une écriture de la division sous la forme

$$f = q_1 g_1 + \cdots + q_s g_s + r.$$

Cette sortie du processus de division (faible ou forte) dépend de l'ordre dans lequel sont faites les récritures, ce que montre l'exemple suivant.

Exemple 23.12 La division de $f = \mathrm{YX}^2 - \mathrm{X}$ par la famille $g_1 = \mathrm{YX} - 1$, $g_2 = \mathrm{X}^2$ pour l'ordre lexicographique induit par $\mathrm{X} < \mathrm{Y}$ peut-être effectuée de deux manières. D'un côté on observe $f \to_{g_1} 0$ et de l'autre $f \to_{g_2} -\mathrm{X}$ et on ne peut plus récrire X. Au sens de la théorie de la récriture, ceci signifie que les règles ne sont pas complètes.

Le point crucial de cette théorie est que les bases standard permettent de caractériser l'appartenance à un idéal :

Proposition 23.9 Si un reste par une division, faible ou forte, d'un dividende f quelconque par une base standard $(g_1,\dots,g_s)$ est nul, le reste par toute autre division faible ou forte l'est aussi. De manière plus générale, l'exposant privilégié du reste par division ne dépend ni de l'ordre des récritures ni de la nature faible ou forte de la division.

Démonstration. Posons $\mathrm{I} = (g_1,\dots,g_s)$. Le procédé de récriture faible ou forte change de représentant mais pas de classe modulo I. Soient r et r' deux restes de divisions faibles ou fortes par $(g_1,\dots,g_s)$. On a $f \equiv r$ et $f \equiv r'$ modulo I, donc $r - r' \in \mathrm{I}$. Si l'un des restes est nul, disons r', alors l'autre, r, est dans I. Comme il n'est pas réductible, il est lui aussi nul, ce qui prouve la première assertion.

Dans le cas contraire, $rr' \neq 0$. Si $\mathrm{mt}(r) \neq \mathrm{mt}(r')$, on peut supposer $\mathrm{mt}(r) > \mathrm{mt}(r')$ auquel cas $\mathrm{mt}(r) = \mathrm{mt}(r - r') \in \mathrm{E(I)}$. Ceci est absurde car on peut alors encore réduire r. Ainsi, r et r' ont même exposant privilégié. ■

Proposition 23.10 Le reste de toute division, faible ou forte, d'un élément f d'un idéal par une base standard de celui-ci est nul.

Démonstration. Le reste est dans l'idéal. S'il n'est pas nul, son monôme de tête est multiple d'un des monômes de tête de la base standard, par définition, ce qui contredit que ce soit un reste. ■

Exemple 23.13 Le théorème d'Apollonius s'obtient par division du polynôme $C = X^2+Y^2-aX-bY$ annulateur du cercle par la base standard (23.2) de l'Exemple 23.11. Comme les polynômes sont de degré 1, le reste de la division est obtenu simplement en remplaçant X par $2ab^2/(a^2+b^2)$ et Y par $2a^2b/(a^2+b^2)$ et il est facile de constater qu'il est nul ; donc C est dans l'idéal engendré par H_1 et H_2 dans $\mathbb{Q}(a,b)[X,Y]$, ce qui conclut la preuve du théorème. (Toute démonstration de théorème de géométrie ne s'obtient pas nécessairement aussi simplement, voir Chapitre 25.)

Proposition 23.11 Le reste d'une division forte d'un dividende f par une base standard $(g_1,\dots,g_s)$ est unique.

Démonstration. Avec les mêmes notations que dans la preuve précédente, si $r-r' \neq 0$ alors $\mathrm{mt}(r-r') \in \mathrm{E(I)}$ provient de l'un des termes de r ou r'. Ceci est absurde car aucun monôme de r ni r' n'est dans E(I) sinon on pourrait le récrire. Donc $r = r'$, ce qui prouve la proposition. ■

Théorème 23.12 — Théorème de division d'Hironaka. Soient I un idéal de R, $<$ un ordre admissible et E(I) l'escalier de I pour cet ordre. Tout polynôme de R est congru modulo I à un unique polynôme appelé reste de la division par l'idéal qui soit est nul, soit a son support à l'extérieur de E(I), c'est-à-dire (strictement) sous l'escalier.

Démonstration. L'opération de division forte par une base standard donne l'existence d'un tel reste. L'unicité est similaire à la preuve de la proposition précédente. ■

La théorie des bases standard permet de prouver ici que tout anneau de polynômes sur un corps possède la propriété de noethérianité, qui consiste à pouvoir engendrer tout idéal par un nombre fini de polynômes.

Corollaire 23.13 — Noethérianité de l'anneau des polynômes. Toute base standard d'un idéal engendre cet idéal.

Démonstration. Le reste de la division d'un élément f de l'idéal par une base standard $(g_1,\dots,g_s)$ ne peut être que nul puisque sinon, son monôme dominant devrait être à la fois à l'extérieur de E(I) (par construction du reste) et dans E(I) (par définition de ce dernier). Il existe donc des quotients q_1, …, q_s tels que $f = q_1g_1 + \cdots + q_sg_s$, et f appartient donc à l'idéal engendré par la base standard. ■

Théorème 23.14 — Décomposition du quotient. En tant que k-espace vectoriel, le quotient R/I est isomorphe au sous-espace vectoriel de R donné comme la somme directe

$$\bigoplus_{a \notin \mathrm{E(I)}} k\mathrm{X}^a.$$

Plus précisement, pour tout entier u, le k-espace vectoriel de R/I engendré par les polynômes homogènes de R de degré u, et noté $(\mathrm{R/I})_u$, est isomorphe à la somme directe $\bigoplus_{a \notin \mathrm{E(I)}, |a|=u} k\mathrm{X}^a$, où $|a|$ représente la somme $a_0 + \cdots + a_n$.

Démonstration. Ce quotient est par définition l'ensemble des classes d'équivalence pour la relation $f \sim g$ quand $f - g \in \mathrm{I}$.

D'une part les restes d'une division forte par une base standard ont leurs supports à l'extérieur de l'escalier E(I). D'autre part, la définition de E(I) assure qu'aucune combinaison linéaire non nulle de monômes à l'extérieur de l'escalier ne peut appartenir à I. Cela prouve le premier isomorphisme.

La dernière assertion du corollaire provient simplement du fait que les divisions faibles et fortes préservent le degré total, sauf en cas de reste nul. ■

Cette décomposition du quotient R/I jouera un rôle théorique important dans le Chapitre 26 sur la dimension des variétés. Aussi, une conséquence pratique majeure concerne le cas où le complémentaire de l'escalier est fini : les monômes à l'extérieur de l'escalier fournissent un moyen d'effectuer facilement des calculs dans R/I avec des algorithmes d'algèbre linéaire. Nous verrons que cette situation, dans le cadre affine, correspond au cas où la variété $\mathbf{V}(\mathrm{I})$ est constituée d'un nombre fini de points.

Finalement, nous avons vu dans ce chapitre que pour effectuer le test à zéro effectivement, il nous suffit de disposer d'une base standard. Le calcul d'une telle base est naturellement le sujet du chapitre suivant.

Notes

Une introduction très facile d'accès et beaucoup plus détaillée des notions développées dans ce chapitre et les quatre suivants est l'ouvrage de Cox, Little et O'Shea [CLO96]. Le livre de Becker et Weispfenning est également très complet d'un point de vue théorique [BW93], et celui d'Eisenbud [Eis95] aborde en détail les aspects algébriques. On peut aussi consulter le livre de Joux couvrant des aspects algorithmiques plus récents, ainsi que des applications à la cryptographie [Jou09]. Le résultat mentionné à la fin de l'Exemple 23.10 est dû à Robbiano [Rob85].

La notion de base standard a été introduite par Hironaka pour calculer des générateurs de l'ensemble des termes initiaux des *fonctions analytiques* au voisinage d'un point [Hir64]. Dans cet ouvrage, nous parlons de base standard pour suggérer cette forme de généralisation. Cependant nous resterons dans le cadre polynomial, ou la terminologie la plus courante est celle des bases de Gröbner.

La terminologie « récriture » utilisée dans ce chapitre et les suivants pour parler des étapes élémentaires des divisions provient de ce qu'on appelle couramment la

théorie de la récriture, développée par Knuth et Bendix à la fin des années soixante dans un contexte informatique lié à la théorie des langages [KB70].

Les définitions adoptées dans cet ouvrage pour les variétés affines et projectives sont restrictives car elles forcent ces variétés à appartenir à un espace affine ou projectif. Néanmoins, elles couvrent bien la plupart des besoins rencontrés en calcul formel. D'une façon informelle, la terminologie habituelle de *variété algébrique* désigne une structure topologique qui s'exprime localement comme une variété affine. Dans ce sens, nos variétés affines et projectives sont des cas particuliers importants de variétés algébriques.

Bibliographie

BW93 Becker, Thomas et Volker Weispfenning (1993). *Gröbner bases. A computational approach to commutative algebra*. Vol. 141. Graduate Texts in Mathematics. Springer-Verlag New York.

CLO96 Cox, David, John Little et Donal O'Shea (1996). *Ideals, varieties, and algorithms. An introduction to computational algebraic geometry and commutative algebra*. 2e éd. Undergraduate Texts in Mathematics. Springer-Verlag.

Eis95 Eisenbud, David (1995). *Commutative algebra. With a view toward algebraic geometry*. Vol. 150. Graduate Texts in Mathematics. Springer-Verlag New York.

Hir64 Hironaka, Heisuke (1964). « Resolution of singularities of an algebraic variety over a field of characteristic zero. I, II ». In : *Annals of Mathematics. Second Series*, vol. 79, p. 205–326.

Jou09 Joux, Antoine (2009). *Algorithmic cryptanalysis*. Chapman & Hall/CRC Cryptography and Network Security Series. Chapman et Hall/CRC.

KB70 Knuth, Donald E. et Peter B. Bendix (1970). « Computational problems in abstract algebra ». In : éd. par John Leech. Pergamon Press. Chap. Simple word problems in universal algebras, p. 263–297.

Rob85 Robbiano, Lorenzo (1985). « Term orderings on the polynomial ring ». In : *EUROCAL '85. European Conference on Computer Algebra. Linz, Austria, April 1-3, 1985. Proceedings. Volume 2 : Research Contributions*. Éd. par Bob F. Caviness. Vol. 204. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, p. 513–517.

24. Construction de bases standard

Résumé

Construire une base standard d'un idéal revient à trouver des générateurs de l'idéal monomial de tête de cet idéal. L'étude des relations entre générateurs d'une présentation de l'idéal donne la manière de compléter la présentation de l'idéal monomial correspondant, et par relèvement de compléter le système de générateurs de l'idéal en une base standard.

24.1 Algorithme naïf par algèbre linéaire

Dans le cadre homogène, un algorithme naïf pour la construction de bases standard ramène les calculs à de l'algèbre linéaire dans des espaces vectoriels de dimension finie, généralisant ainsi par exemple les calculs de résultant via la matrice de Sylvester. Soit $\mathrm{I} = (f_1, \ldots, f_s)$ un idéal homogène de $\mathrm{R} = k[\mathrm{X}_0, \ldots, \mathrm{X}_n]$, avec f_i homogène de degré d_i, et pour tout d soit R_d l'ensemble des polynômes homogènes de degré d dans R, qui est un k-espace vectoriel de dimension finie. Pour $u \in \mathbb{N}$, l'application

$$\phi_u : \left\{ \begin{array}{ccc} \mathrm{R}_{u-d_1} \times \cdots \times \mathrm{R}_{u-d_s} & \to & \mathrm{R}_u \\ (g_1, \ldots, g_s) & \mapsto & \sum_i g_i f_i \end{array} \right.$$

a pour image l'espace vectoriel I_u défini en Proposition 23.3. Pour un ordre admissible $<$ donné et pour un degré v, soit B_v la base de R_v formée des monômes homogènes de degré v ordonnés par $<$, de sorte que R_v s'identifie à un espace vectoriel de vecteurs colonnes. Les colonnes de la matrice de ϕ_u dans ces bases engendrent alors I_u comme k-espace vectoriel. Cette matrice s'appelle la *matrice de Macaulay*. Par opérations élémentaires sur ses colonnes, on peut la mettre sous forme échelonnée (voir Chapitre 8). Les colonnes de la nouvelle matrice engendrent bien sûr encore I_u, et les premiers

coefficients non nuls des colonnes donnent les monômes de tête des polynômes de I_u. On en déduit le résultat de complexité suivant.

Proposition 24.1 Étant donnés $f_1,\dots,f_s$ des polynômes homogènes appartenant à $k[X_0,\dots,X_n]$ et un entier D, on peut calculer les éléments de degré au plus D d'une base standard de $I=(f_1,\dots,f_s)$ en

$$O\left(sD\binom{n+D}{D}^{\theta}\right)$$

opérations arithmétiques dans k, où θ est un exposant réalisable pour la multiplication de matrices.

Démonstration. La construction des matrices de Macaulay pour les degrés u de 1 à D n'utilise pas d'opération arithmétique. La dimension de R_u est égale au nombre de monômes homogènes de degré u en $n+1$ variables, c'est-à-dire $\binom{n+u}{u}$. En degré u, la matrice de Macaulay a donc dimensions

$$\left(\binom{n+u-d_1}{u-d_1}+\cdots+\binom{n+u-d_s}{u-d_s}\right)\times\binom{n+u}{u}.$$

La suite $\binom{n+u}{u}$ est croissante avec u, de sorte que le nombre de lignes est borné par $s\binom{n+u}{u}$. Par les résultats du Chapitre 8, le calcul de la forme échelonnée peut être effectué en

$$O\left(s\binom{n+u}{u}^{\theta}\right)$$

opérations arithmétiques dans k. La somme de ces calculs pour $u=0,\dots,D$ donne le résultat de complexité de la proposition. De ces formes échelonnées, on extrait ensuite la base standard, sans calcul supplémentaire. ■

En effectuant ce calcul pour D assez grand, par noethérianité, la réunion des colonnes obtenues forme une base standard de l'idéal. Pour transformer cette idée en algorithme il faut disposer d'une borne supérieure du degré maximal D à atteindre. Une telle borne existe mais est disproportionnée en général. Les chapitres suivants présentent néanmoins des familles de systèmes pour lesquels les bornes ainsi obtenues sont réalistes, et mènent à un calcul en bonne complexité.

24.2 Polynôme de syzygie

Nous revenons au cas général inhomogène. L'algorithme de la section précédente ne considère que la structure d'espace vectoriel de I, mais n'exploite pas sa stabilité par produit. Du point de vue des R-modules, le noyau de la même application ϕ_u pour u assez grand est par définition le *module des relations* entre $f_1,\dots,f_s$, ou *(premier) module de syzygies*. On peut le construire par algèbre linéaire sur k en degré donné.

Il se trouve que le module des relations entre les éléments d'une base standard se construit facilement ; en retour ceci conduit à un algorithme de construction d'une base standard à partir d'un système donné de générateurs.

Définition 24.1 Soit $(f_1,\dots,f_s)$ une famille de polynômes non nuls. Pour i différent de j, le *polynôme de syzygie*, ou *S-polynôme*, est le polynôme

$$\mathrm{S}(f_i,f_j)=\frac{\mathrm{tt}(f_j)f_i-\mathrm{tt}(f_i)f_j}{\mathrm{pgcd}\Big(\mathrm{mt}(f_i),\mathrm{mt}(f_j)\Big)}. \tag{24.1}$$

Sous l'hypothèse additionnelle que les f_i constituent une base standard, en divisant, faiblement ou fortement, $\mathrm{S}(f_i,f_j)$ par $(f_1,\dots,f_s)$, on obtient un reste nul et donc une relation entre les polynômes de la base standard ; nous conviendrons d'appeler une telle relation une *relation bilatérale évidente*. Elle est de la forme $\mathrm{S}(f_i,f_j)=q_1f_1+\cdots+q_sf_s$, avec pour chaque ℓ entre 1 et s, $\mathrm{mt}(q_\ell f_\ell)<\mathrm{mt}\Big(\mathrm{mt}(f_j)\,f_i\Big)=\mathrm{mt}\Big(\mathrm{mt}(f_i)\,f_j\Big)$.

Exemple 24.1 La base standard $g_1=2\mathrm{XY}-\mathrm{XZ}, g_2=\mathrm{X}^2, g_3=\mathrm{X}^2\mathrm{Z}$ pour l'ordre 1ex induit par $\mathrm{Z}<\mathrm{Y}<\mathrm{X}$ donne les relations bilatérales évidentes : $\mathrm{X}g_1-2\mathrm{Y}g_2=-g_3$, $\mathrm{XZ}g_1-2\mathrm{Y}g_3=-\mathrm{Z}g_3$, $\mathrm{Z}g_2-g_3=0$.

Une première remarque est que le choix de normalisation du coefficient dominant du polynôme (24.1) n'a pas d'incidence sur la théorie, vu que seul les espaces vectoriels engendrés comptent. Par ailleurs, une définition équivalente des polynômes de syzygie est

$$\mathrm{S}(f_i,f_j)=c_j\frac{\mathrm{ppcm}\Big(\mathrm{mt}(f_i),\mathrm{mt}(f_j)\Big)}{\mathrm{mt}(f_i)}f_i-c_i\frac{\mathrm{ppcm}\Big(\mathrm{mt}(f_i),\mathrm{mt}(f_j)\Big)}{\mathrm{mt}(f_j)}f_j,$$

après avoir posé $\mathrm{tt}(f_i)=c_i\,\mathrm{mt}(f_i)$ et $\mathrm{tt}(f_j)=c_j\,\mathrm{mt}(f_j)$. Cette formulation est le calcul effectué en pratique. L'ensemble des relations bilatérales évidentes n'est pas aussi particulier qu'on pourrait le penser, en raison du théorème suivant.

Théorème 24.2 Les relations bilatérales évidentes engendrent le module des relations entre les éléments d'une base standard.

Démonstration. Si $\sum_{i=1}^{s}g_if_i=0$ est une relation entre les générateurs, soient $m_i=\mathrm{mt}(g_if_i)$, m le maximum des m_i pour $<$ et T l'ensemble des indices i tels que m_i soit égal à m. L'ensemble T ne peut pas être réduit à un seul élément. Soient r et t deux éléments distincts de T ; m est un multiple commun de $\mathrm{mt}(f_r)$ et $\mathrm{mt}(f_t)$, donc du ppcm de ces deux monômes, et à l'aide d'un multiple de la relation bilatérale évidente entre f_r et f_t, on peut récrire la relation initiale en une nouvelle pour laquelle soit T contient moins d'éléments, soit le monôme m est plus petit. Il suffit alors de réappliquer le procédé aussi longtemps que possible pour conclure. ■

24.3 L'algorithme de construction de Buchberger

L'algorithme de Buchberger, s'inspirant de ce qui précède, construit une base standard de I à partir d'un système donné F de générateurs, en procédant par adjonctions successives de restes non nuls de divisions de polynômes de syzygies. Cette méthode

Entrée Un système de générateurs F de l'idéal I.
Sortie Une base standard G de l'idéal I.
1. Initialiser G avec F.
2. Répéter :
   a. initialiser G' avec G ;
   b. pour toute paire p, q ($p \neq q$) de G' faire :
      i. diviser $S(p,q)$ par G',
      ii. si le reste r obtenu est non nul alors remplacer G par $G \cup \{r\}$;
   jusqu'à $G = G'$.

Algorithme 24.1 – Calcul d'une base standard par l'algorithme de Buchberger.

est détaillée dans l'Algorithme 24.1, la division utilisée est, au choix, faible ou forte. La correction de l'algorithme repose sur le résultat suivant.

Théorème 24.3 Une famille de polynômes $\{g_1, \ldots, g_r\}$ est une base standard de l'idéal qu'ils engendrent si, et seulement si, pour tout $1 \leq i, j \leq r$ le polynôme de syzygie $S(g_i, g_j)$ se réduit à zéro après division faible par $\{g_1, \ldots, g_r\}$.

Démonstration. L'implication directe est conséquence de la Proposition 23.10 du chapitre précédent. La réciproque est de même nature que la preuve du théorème de Spear–Schreyer : soit $f = \sum_i f_i g_i$ un polynôme non nul de l'idéal, et montrons que son monôme de tête est dans l'idéal monomial engendré par $\mathrm{mt}(g_1), \ldots, \mathrm{mt}(g_r)$. Pour cela, soient $m_i = \mathrm{mt}(g_i f_i)$, m le maximum des m_i, T l'ensemble des indices i tels que m_i soit égal à m. Si $m \neq \mathrm{mt}(f)$, alors T a au moins deux éléments r et t, et on utilise la récriture donnée par $S(g_r, g_t)$ pour diminuer T jusqu'à ce que $m = \mathrm{mt}(f)$ comme suit. Notons $m_{r,t}$ le ppcm de $\mathrm{mt}(g_r)$ et $\mathrm{mt}(g_t)$, de sorte que

$$S(g_r, g_t) = \mathrm{ct}(g_t)\frac{m_{r,t}}{\mathrm{mt}(g_r)} g_r - \mathrm{ct}(g_r)\frac{m_{r,t}}{\mathrm{mt}(g_t)} g_t.$$

On obtient alors l'égalité suivante :

$$f_r g_r + f_t g_t - \frac{\mathrm{ct}(f_r)m}{\mathrm{ct}(g_t)m_{r,t}} S(g_r, g_t) = (f_r - \mathrm{tt}(f_r))g_r + \left(f_t + \frac{\mathrm{ct}(g_r)\,\mathrm{ct}(f_r)m}{\mathrm{ct}(g_t)\,\mathrm{mt}(g_t)}\right) g_t.$$

Comme $\frac{\mathrm{ct}(f_r)m}{\mathrm{ct}(g_t)m_{r,t}} S(g_r, g_t)$ est divisible par $g_1, \ldots, g_r$ et ne contient que des monômes strictement inférieurs à m, alors on obtient une nouvelle écriture de la forme $f = \sum_i f_i g_i$ avec un ensemble T strictement plus petit.

En itérant le procédé, on finit bien par trouver une écriture telle que $\mathrm{mt}(f)$ soit multiple d'un $\mathrm{mt}(g_i)$. Les $\mathrm{mt}(g_i)$ engendrent donc l'escalier de l'idéal. Par la Définition 23.22 du chapitre précédent, $\{g_1, \ldots, g_r\}$ est une base standard. ■

Corollaire 24.4 L'algorithme de Buchberger termine et est correct.

Démonstration. Pour la terminaison, il suffit de considérer l'idéal monomial mt(G) engendré par $\mathrm{mt}(g_1),\dots,\mathrm{mt}(g_s)$. À chaque tour de la boucle l'idéal croît au sens large. Il est constant à partir d'un certain rang par noethérianité. Or, si mt(G) stagne, G stagne. En effet un reste r non nul agrandirait l'idéal monomial. Autrement son monôme de tête serait réductible et la division continuerait après r.

La correction de l'algorithme de Buchberger est un corollaire du théorème précédent : si l'algorithme s'arrête, c'est que tous les polynômes de syzygies se réduisent à zéro. ■

Exercice 24.1 — Critères de Buchberger. Les deux critères suivants permettent de déterminer sans calcul que certains S-polynômes se réduisent à zéro, ce qui permet d'accélérer l'algorithme de Buchberger. Soient $\mathrm{R} = k[\mathrm{X}_1\dots,\mathrm{X}_n]$ et $<$ un ordre monomial sur R. Montrer les deux critères suivants :

1. *Premier critère.* Soient $f,g \in \mathrm{R}$ tels que leurs monômes de tête n'aient aucune variable en commun. Alors le reste de la division, faible ou forte, de $\mathrm{S}(f,g)$ par la famille $\{f,g\}$ est nul.
2. *Second critère.* Soient $\mathrm{F} \subset \mathrm{R}$ une famille finie de R, $p,f,g,\in \mathrm{R}$ des polynômes tels que :
   a. $\mathrm{mt}(p) \mid \mathrm{ppcm}\big(\mathrm{mt}(f),\mathrm{mt}(g)\big)$;
   b. les restes de la division forte de $\mathrm{S}(f,p)$ et $\mathrm{S}(g,p)$ par F sont nuls.

   Alors le reste de la division forte de $\mathrm{S}(f,g)$ par F est nul.

■

24.4 Exemple de calcul

Nous détaillons ici le calcul de la base standard de l'idéal $(\mathrm{X}+\mathrm{Y}-\mathrm{Z}, \mathrm{X}^2-2\mathrm{T}^2, \mathrm{Y}^2-5\mathrm{T}^2)$ pour l'ordre lexicographique induit par $\mathrm{T}<\mathrm{Z}<\mathrm{Y}<\mathrm{X}$. Nous présentons tous les polynômes du calcul par monômes décroissants, en soulignant le monôme de tête. Dans ce qui suit, nous ajoutons une règle de récriture dont le lecteur se convaincra qu'elle ne change rien à la théorie, en se permettant de remplacer un polynôme par tout multiple non nul par un entier, la fonction de cette récriture étant d'éviter l'apparition de dénominateurs. Nous utiliserons le symbole $\sim$ pour une telle récriture vide.

Le point de départ est la famille :

$$p_1 := \underline{\mathrm{X}}+\mathrm{Y}-\mathrm{Z}, \qquad p_2 := \underline{\mathrm{X}^2}-2\mathrm{T}^2, \qquad p_3 := \underline{\mathrm{Y}^2}-5\mathrm{T}^2.$$

Par le premier critère de Buchberger, $\mathrm{S}(p_1,p_3)$ et $\mathrm{S}(p_2,p_3)$ vont se réduire à zéro. Il suffit donc de réduire

$$\begin{aligned}\mathrm{S}(p_1,p_2) &= \mathrm{X}p_1 - p_2 = \underline{\mathrm{XY}} - \mathrm{XZ} + 2\mathrm{T}^2 \\ &\to_{p_1} -\underline{\mathrm{XZ}} - \mathrm{Y}^2 + \mathrm{YZ} + 2\mathrm{T}^2 \to_{p_1} -\underline{\mathrm{Y}^2} + 2\mathrm{YZ} - \mathrm{Z}^2 + 2\mathrm{T}^2 \to_{p_3} 2\underline{\mathrm{YZ}} - \mathrm{Z}^2 - 3\mathrm{T}^2,\end{aligned}$$

ce qui introduit un nouveau polynôme dans la base standard en construction :

$$p_4 := 2\underline{YZ} - Z^2 - 3T^2.$$

Comme précédemment, les polynômes de syzygies $S(p_1,p_4)$ et $S(p_2,p_4)$ vont se réduire à zéro, si bien que nous ne considérons que la réduction suivante, qui mène à un nouveau polynôme p_5 :

$$\begin{aligned} S(p_3,p_4) = 2Zp_3 - Yp_4 = \underline{YZ^2} + 3YT^2 - 10ZT^2 \\ \sim 2\underline{YZ^2} + 6YT^2 - 20ZT^2 \to_{p_4} 6\underline{YT^2} - 17ZT^2 + Z^3 =: p_5. \end{aligned}$$

Ne subsistent ensuite que deux réductions intéressantes, celles de $S(p_3,p_5)$ et $S(p_4,p_5)$:

$$\begin{aligned} S(p_3,p_5) = 6T^2p_3 - Yp_5 = -\underline{YZ^3} + 17YZT^2 - 30T^4 \sim -2\underline{YZ^3} + 34YZT^2 - 60T^4 \\ \to_{p_4} 34\underline{YZT^2} - Z^4 - 3Z^2T^2 + 60T^4 \to_{p_4} -\underline{Z^4} + 14Z^2T^2 - 9T^4 := p_6, \\ S(p_4,p_5) = 3T^2p_4 - Zp_5 = -\underline{Z^4} + 14Z^2T^2 - 9T^4 \to_{p_6} 0, \end{aligned}$$

après lesquelles p_6 ne produit que des polynômes de syzygie qui se réduisent à 0.

Ainsi, $G_1 = \{p_1,\dots,p_6\}$ est une base standard de l'idéal considéré. Cette base n'est pas minimale, puisque le monôme de tête de p_2 est divisible par celui de p_1, mais $G_2 = \{p_1,p_3,\dots,p_6\}$ l'est : en effet, les deux systèmes engendrent le même idéal et comme nous n'avons jamais utilisé p_2 dans les divisions ci-dessus, les polynômes de syzygie de G_2 se réduisent tous à 0 par G_2, qui est donc aussi une base standard.

Exercice 24.2

1. Déduire des calculs précédents que $\sqrt{2}+\sqrt{5}$ est algébrique sur le corps des rationnels $\mathbb{Q}$, en exhibant un polynôme à une variable à coefficients rationnels dont il est racine.
2. Quel est le résultant de $(Y-Z)^2-2$ et Y^2-5 par rapport à Y ?
3. Déduire des calculs précédents que $\mathbb{Q}(\sqrt{2},\sqrt{5}) = \mathbb{Q}(\sqrt{2}+\sqrt{5})$. Exprimer $\sqrt{2}$ et $\sqrt{5}$ en fonction de $\sqrt{2}+\sqrt{5}$.

■

24.5 Propriétés des bases standard pour quelques ordres

Les ordres `lex` et `tdeg` ont des propriétés distinctes, qui mènent à des usages différents.

Ordre `lex` — Projection et implicitation

Nous rappelons que `lex` est l'ordre lexicographique induit par $X_n <_{lex} X_{n-1} <_{lex} \cdots <_{lex} X_1$.

Proposition 24.5 Soit $g \in k[X_1,\dots,X_n]$. Si X_1 ne divise pas le monôme de tête pour `lex` de g alors $g \in k[X_2,\dots,X_n]$.

Démonstration. Le monôme de tête est l'un des monômes qui contient le plus de X_1. Si un monôme de g contient du X_1 alors son monôme de tête en contient. ■

Théorème 24.6 Soit $g_1,\dots,g_u,g_{u+1},\dots,g_s$ une base standard d'un idéal I pour `lex`. Supposons que X_1 divise les monômes de tête de $g_1,\dots,g_u$ uniquement. Alors la famille $g_{u+1},\dots,g_s$ engendre $I\cap k[X_2,\dots,X_n]$ et en forme une base standard.

Démonstration. D'après la proposition précédente, $g_{u+1},\dots,g_s \in k[X_2,\dots,X_n]$ donc $(g_{u+1},\dots,g_s)\subseteq I\cap k[X_2,\dots,X_n]$. Soit $f\in I\cap k[X_2,\dots,X_n]$. Le terme monôme tête $\mathrm{mt}(f)$ est dans l'escalier engendré par $\mathrm{mt}(g_1),\dots,\mathrm{mt}(g_s)$ donc c'est un multiple d'un $\mathrm{mt}(g_i)$ pour un certain $i\in\{1,\dots,s\}$. On a nécessairement $i\notin\{1,\dots,u\}$ car X_1 ne divise pas $\mathrm{mt}(f)$. Nous venons de montrer que $g_{u+1},\dots,g_s$ est une base standard de $I\cap k[X_2,\dots,X_n]$ et donc que cette famille l'engendre. ■

Le pendant géométrique de l'élimination de X_1 dans l'idéal I est une projection selon X_1 sur les autres coordonnées. Nous prouverons ce résultat dans le chapitre suivant qui donnera plus d'outils pour faire le lien entre l'algèbre et la géométrie.

Par l'une ou l'autre des interprétations, on observe qu'un polynôme de I qui ne fait pas intervenir X_1 fournit une équation vérifiée par les coordonnées $(x_2,\dots,x_n)$ d'un zéro x des polynômes de I *quelle que soit la valeur de sa coordonnée* x_1. L'élimination est donc bien l'outil pour répondre au problème d'implicitation donné en motivation au début du chapitre précédent.

Exercice 24.3 Soient $I=(f_1,\dots,f_s)$ et $J=(g_1,\dots,g_r)$ deux idéaux de $k[X_1,\dots,X_n]$. Soit T une nouvelle variable et soit

$$K := (Tf_1,\dots,Tf_s,(1-T)g_1,\dots,(1-T)g_r)$$

vu comme un idéal de $k[T,X_1,\dots,X_n]$.

1. Montrer que $I\cap J = K\cap k[X_1,\dots,X_n]$.
2. En déduire un algorithme pour calculer un ensemble de polynômes générateurs de l'intersection $I\cap J$.

■

Exercice 24.4 Montrer que le Théorème 24.6 se généralise au cas des ordres d'élimination introduits à l'Exemple 23.9 : les polynômes d'une base standard de I pour le i^{e} ordre d'élimination qui ne font pas intervenir les variables $X_1,\dots,X_i$ forment une base standard de $I\cap k[X_{i+1},\dots,X_n]$. ■

Ordre `tdeg` en homogène — Section par un hyperplan

Nous rappelons que l'ordre `tdeg` en les variables $X_0,\dots,X_n$ consiste à ordonner les monômes par degré total, puis les plus grands sont ceux qui ont le moins de X_n, puis le moins de X_{n-1} et ainsi de suite (Définition 23.17).

Proposition 24.7 Soit g un polynôme homogène de $k[X_0,\dots,X_n]$. Si X_n divise le monôme de tête de g pour `tdeg`, alors X_n divise g.

Démonstration. Comme g est homogène, son monôme de tête est l'un des monômes contenant le moins de X_n. S'il y a du X_n dans le monôme de tête de g, il y en a dans tous les autres monômes et ainsi X_n divise g. ■

Théorème 24.8 Soit $g_1,\dots,g_u,g_{u+1},\dots,g_s$ une base standard homogène d'un idéal I pour `tdeg` ordonnée telle que X_n divise uniquement les monômes de tête de $g_{u+1},\dots,g_s$. Alors la famille $g_1,\dots,g_u,X_n$ engendre $I+(X_n)$ et en forme une base standard.

Démonstration. D'après la proposition précédente, X_n divise $g_{u+1},\dots,g_s$. On en déduit facilement que $(g_1,\dots,g_u,X_n)=I+(X_n)$. Soit $f\in I+(X_n)$. Montrons que son monôme de tête $\mathrm{mt}(f)$ est un multiple de l'un des monômes de tête $\mathrm{mt}(g_1),\dots,\mathrm{mt}(g_u)$, ou de X_n. Si X_n divise le monôme de tête $\mathrm{mt}(f)$ de f alors c'est bien le cas. Sinon, soit $h\in(X_n)$ tel que $f-h\in I$. On vérifie que $\mathrm{mt}(f)=\mathrm{mt}(f-h)$ et donc $\mathrm{mt}(f)$ est multiple de l'un des $\mathrm{mt}(g_i)$ pour $i\in\{1,\dots,s\}$. Cependant, $i\notin\{u+1,\dots,s\}$, car X_n ne divise pas $\mathrm{mt}(f)$. Nous venons de montrer que $(g_1,\dots,g_u,X_n)$ est une base standard de l'idéal $I+(X_n)$. ■

L'ajout de l'équation X_n à l'idéal I correspond donc à l'opération d'intersection de $\mathbf{V}(I)$ avec l'hyperplan X_n.

Notes

L'Algorithme 24.1 remonte à la thèse de Buchberger en 1965 [Buc65] qui l'a exprimé dans le cas affine, en appelant le résultat *base de Gröbner* (du nom de son directeur de thèse). Des preuves détaillées de l'algorithme de Buchberger et du Théorème 24.2 se trouvent dans de livre de Cox, Little et O'Shea [CLO96, Section 2.6, Theorem 6 ; Section 2.7].

L'étude des propriétés du module des syzygies d'un idéal a été réalisée indépendamment par Spear [Spe78] et Schreyer [Sch80a], et a mené au Théorème 24.2, ainsi qu'à un algorithme de calcul d'un système de générateurs de ce module. Ces résultats trouvent néanmoins leurs racines dans les travaux de Janet [Jan20]. Cette approche algorithmique a pu être ensuite prolongée par Möller et Mora [MM86]. Pour davantage de détails nous renvoyons le lecteur à l'ouvrage de Mora [Mor05, Chapter 23.8].

L'ordre monomial utilisé a une influence capitale sur le coût des calculs en pratique. En général les ordres d'élimination et en particulier l'ordre lexicographique sont les plus coûteux. Il est alors intéressant de calculer une base pour un ordre moins coûteux, puis d'utiliser un algorithme de changement d'ordre. De premiers éléments de comparaison ont été donnés par Lazard [Laz83]. Pour les systèmes admettant un nombre fini de solutions, le changement d'ordre est une question « d'algèbre linéaire », et un algorithme efficace a été proposé par Faugère, Gianni, Lazard et Mora [Fau+93]. Pour la dimension positive, la technique désormais classique des « marches de Gröbner » (*Gröbner walk*, en anglais) est due à Collart, Kalkbrenner et Mall [CKM97].

Le choix des paires de polynômes p,q de l'Algorithme 24.1, souvent appelées paires critiques, pour former de nouveaux éléments de la base, est crucial car de nombreuses paires se réduisent en général à zéro. Les critères de Buchberger peuvent être complétés et nous renvoyons à l'ouvrage de Becker et Weispfenning [BW93] pour davantage de précisions. Une stratégie pratique est d'essayer de faire comme si la situation était homogène [Gio+91]. Lorsque le corps de base est celui des nombres rationnels, il est aussi possible de faire les calculs modulo un nombre premier de la

taille d'un mot machine et de conserver la trace du calcul pour prévoir quelles sont les paires critiques qui se réduisent à zéro [Tra89]. Enfin l'algorithme F5 de Faugère permet d'éliminer les réductions à zéro inutiles lorsque la suite de polynômes en entrée est régulière [Fau02].

D'un point de vue pratique les structures de données utilisées pour représenter les polynômes à plusieurs variables jouent un rôle important. Mais depuis l'algorithme F4, proposé par Faugère en 2004, les meilleurs implémentations construisent plutôt des matrices de Macaulay partielles et font appel à des techniques d'algèbre linéaire creuse [Fau99] plutôt que de calculer directement sur les polynômes.

Les questions de complexité du calcul de bases standard sont abordées au Chapitre 26.

Bibliographie

Buc65 Buchberger, Bruno (1965). « Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal ». Thèse de doctorat. Mathematical Institute, University of Innsbruck, Austria.

BW93 Becker, Thomas et Volker Weispfenning (1993). *Gröbner bases. A computational approach to commutative algebra*. Vol. 141. Graduate Texts in Mathematics. Springer-Verlag New York.

CKM97 Collart, S., M. Kalkbrener et D. Mall (1997). « Converting bases with the Gröbner walk ». In : *Journal of Symbolic Computation*, vol. 24, n°3-4, p. 465–469.

CLO96 Cox, David, John Little et Donal O'Shea (1996). *Ideals, varieties, and algorithms. An introduction to computational algebraic geometry and commutative algebra*. 2e éd. Undergraduate Texts in Mathematics. Springer-Verlag.

Fau+93 Faugère, J. C., P. Gianni, D. Lazard et T. Mora (1993). « Efficient computation of zero-dimensional Gröbner bases by change of ordering ». In : *Journal of Symbolic Computation*, vol. 16, n°4, p. 329–344.

Fau02 Faugère, Jean-Charles (2002). « A new efficient algorithm for computing Gröbner bases without reduction to zero (F5) ». In : *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*. ISSAC'02. New York, NY, USA : ACM, p. 75–83.

Fau99 — (1999). « A new efficient algorithm for computing Gröbner bases (F4) ». In : *Journal of Pure and Applied Algebra*, vol. 139, n°1–3, p. 61–88.

Gio+91 Giovini, Alessandro, Teo Mora, Gianfranco Niesi, Lorenzo Robbiano et Carlo Traverso (1991). « "One sugar cube, please" or selection strategies in the Buchberger algorithm ». In : *Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation*. ISSAC'91. New York, NY, USA : ACM, p. 49–54.

Jan20 Janet, Maurice (1920). « Sur les systèmes d'équations aux dérivées partielles ». In : *Journal de Mathématiques Pures et Appliquées*, vol. 3, p. 65–151.

Laz83 Lazard, D. (1983). « Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations ». In : *Computer Algebra : EUROCAL'83, European Computer Algebra Conference London, England, March 28–30, 1983 Proceedings*. Éd. par J. A. Hulzen. Springer Berlin Heidelberg, p. 146–156.

MM86 Möller, H. Michael et Ferdinando Mora (1986). « New constructive methods in classical ideal theory ». In : *Journal of Algebra*, vol. 100, n°1, p. 138–178.

Mor05 Mora, Teo (2005). *Solving polynomial equation systems II : Macaulay's paradigm and Gröbner technology*. Vol. 99. Encyclopedia of Mathematics and its Applications. Cambridge University Press.

Sch80a Schreyer, Frank-Olaf (1980). « Die Berechnung Syzygien mit dem verallgemeinerten Weierstrasschen Divisionsatz ». Diplomarbeit. Hamburg.

Spe78 Spear, D. (1978). « A constructive approach to commutative ring theory ». In : *Processing of the 1977 MACSYMA Users' Conference (NASA)*, p. 369–376.

Tra89 Traverso, Carlo (1989). « Gröbner trace algorithms ». In : *Symbolic and Algebraic Computation : International Symposium ISSAC'88 Rome, Italy, July 4–8, 1988 Proceedings*. Éd. par P. Gianni. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 125–138.

25. Nullstellensatz et applications

Résumé

Le Nullstellensatz (en allemand, « théorème du lieu des zéros ») est le théorème qui fait le lien entre les variétés algébriques et les idéaux. Il en découle que certaines opérations géométriques se traduisent aisément en algorithmes sur les équations. Ce théorème permet en particulier de caractériser les projections des variétés affines et projectives, et de définir une représentation confortable des variétés affines ayant un nombre fini de points.

25.1 Nullstellensatz affine

Le résultat de cette section est un lien explicite entre l'algèbre et la géométrie, à l'aide d'une nouvelle notion : le radical d'un idéal. On continue de noter R l'anneau de polynômes $k[X_1, \ldots, X_n]$.

Définition 25.1 Soit I un idéal de R. Le *radical* de I, noté $\sqrt{I}$, est l'idéal $\{f \in R \mid \exists e \in \mathbb{N},\ f^e \in I\}$; il contient I. Un idéal I est dit *radical* si $I = \sqrt{I}$.

Théorème 25.1 — Nullstellensatz affine fort. Si k est algébriquement clos, et si I est un idéal de R, alors tout polynôme $g \in R$ qui s'annule identiquement sur $\mathbf{V}(I)$ appartient au radical $\sqrt{I}$ de I.

La preuve de ce théorème choisie ici repose sur l'utilisation de variables particulières.

Changement des variables

Proposition 25.2 Si k est infini, alors pour tout $f \in \mathrm{R}$ non nul, il existe des éléments $a_1, \ldots, a_{n-1}$ dans k tels que le changement de variables inversible

$$\begin{cases} \mathrm{X}_1 & \leftarrow & \mathrm{X}_1 + a_1 \mathrm{X}_n \\ \mathrm{X}_2 & \leftarrow & \mathrm{X}_2 + a_2 \mathrm{X}_n \\ \vdots & \leftarrow & \vdots \\ \mathrm{X}_{n-1} & \leftarrow & \mathrm{X}_{n-1} + a_{n-1} \mathrm{X}_n \end{cases} \tag{25.1}$$

transforme f en le produit d'un polynôme unitaire en X_n par une constante non nulle de k.

Démonstration. Soit d le degré de f et f_d la partie de degré d de f. Alors le coefficient de tête de $f(\mathrm{X}_1 + a_1\mathrm{X}_n, \mathrm{X}_2 + a_2\mathrm{X}_n, \ldots, \mathrm{X}_{n-1} + a_{n-1}\mathrm{X}_{n-1}, \mathrm{X}_n)$ vu comme polynôme en X_n est $f_d(a_1, \ldots, a_{n-1}, 1)$ que l'on peut prendre non nul par le lemme suivant. ■

Lemme 25.3 Si k est infini et si $f \in \mathrm{R}$ est non nul, alors $k^n \setminus \mathbf{V}(f)$ est non vide.

Démonstration. La preuve s'effectue par récurrence sur le nombre de variables. Si $n = 1$ le lemme est vrai. Supposons qu'il soit vrai pour $n \geq 1$. En appliquant l'hypothèse de récurrence, il existe $(a_1, \ldots, a_{n-1}) \in k^{n-1}$ qui n'annule pas le terme constant de f vu comme polynôme en X_n à coefficients dans $k[\mathrm{X}_1, \ldots, \mathrm{X}_{n-1}]$. Ainsi $f(a_1, \ldots, a_{n-1}, \mathrm{X}_n)$ est un polynôme non nul. Il a donc un nombre fini de racines. ■

Nullstellensatz faible

Théorème 25.4 — Nullstellensatz affine faible. Si k est algébriquement clos, et si I est un idéal de R tel que $\mathbf{V}(\mathrm{I}) = \varnothing$, alors $\mathrm{I} = \mathrm{R}$.

Exemple 25.1 Il est nécessaire de regarder les zéros sur un corps k algébriquement clos. En effet soit $\mathrm{I} = (\mathrm{X}^2 + 1)$. Alors $\mathbf{V}_{\mathbb{R}}(\mathrm{I}) = \varnothing$ mais $1 \notin \mathrm{I}$. Ceci est cohérent avec le résultat précédent car la variété $\mathbf{V}_{\mathbb{C}}(\mathrm{I})$ se réduit aux points complexes $\pm i$.

Démonstration. Si I est un idéal de R, par noetherianité (Corollaire 23.13), il existe un nombre fini de polynômes $f_1, \ldots, f_s$ tels que $\mathrm{I} = (f_1, \ldots, f_s)$. Si $\mathrm{I} \neq (1)$, c'est-à-dire que I est un idéal propre de R alors il s'agit de prouver que $\mathbf{V}(\mathrm{I}) \neq \varnothing$. La preuve se fait par récurrence sur la dimension n de l'espace ambiant. Si $n = 1$, alors comme $k[\mathrm{X}_1]$ est principal, $\mathrm{I} = (f)$ avec $f \neq 1$, et f a une racine, car k est algébriquement clos.

Supposons le résultat vérifié en dimension $n-1$. Quitte à faire le changement de coordonnées de la Proposition 25.2, on peut supposer f_1 unitaire en X_n (un corps algébriquement clos est nécessairement infini). On pose $\mathrm{J} = \mathrm{I} \cap k[\mathrm{X}_1, \ldots, \mathrm{X}_{n-1}]$. Comme

J ne contient pas 1, il est propre. Par récurrence, il existe donc $(a_1,\dots,a_{n-1}) \in \mathbf{V}(\mathrm{J})$. On considère ensuite l'idéal

$$\mathrm{H} := \{f(a_1,\dots,a_{n-1},\mathrm{X}_n) \mid f \in \mathrm{I}\}$$

de $k[\mathrm{X}_n]$. Si cet idéal est propre, alors il y a un zéro dans $\mathbf{V}(\mathrm{H})$ et donc un zéro dans $\mathbf{V}(\mathrm{I})$. Il ne reste donc plus qu'à montrer que pour tout f non nul de I, $f(a_1,\dots,a_{n-1},\mathrm{X}_n)$ n'est pas constant. Pour $f \in \mathrm{I}$, soit $g := \mathrm{Res}_{\mathrm{X}_n}(f_1, f)$. Par la Proposition 6.7, g appartient à $\mathrm{I} \cap k[\mathrm{X}_1,\dots,\mathrm{X}_{n-1}]$ et donc $g(a_1,\dots,a_{n-1}) = 0$. La Proposition 6.8 s'applique parce que f_1 est unitaire, et entraîne que

$$g(a_1,\dots,a_{n-1}) = \mathrm{Res}_{\mathrm{X}_n}(f_1(a_1,\dots,a_{n-1},\mathrm{X}_n), f(a_1,\dots,a_{n-1},\mathrm{X}_n)),$$

ce qui implique que $f(a_1,\dots,a_{n-1},\mathrm{X}_n)$ n'est pas constant. ■

Astuce de Rabinowitsch

Proposition 25.5 Supposons k algébriquement clos, et soient $\mathrm{I} = (f_1, \dots, f_s)$ un idéal de R et g un polynôme qui s'annule identiquement sur $\mathbf{V}(\mathrm{I})$. Alors la variété de l'idéal $\mathrm{I}' := \mathrm{I} + (\mathrm{Y}g - 1)$ de $k[\mathrm{X}_1,\dots,\mathrm{X}_n,\mathrm{Y}]$ est vide.

Démonstration. D'une part, $\mathrm{I} \subset \mathrm{I}'$ implique $\mathbf{V}(\mathrm{I}') \subset \mathbf{V}(\mathrm{I})$ dans $\mathbb{A}^{n+1}$. Mais d'autre part, puisque $y\,g(x_1,\dots,x_n) - 1 = 0$ implique $g(x_1,\dots,x_n) \neq 0$, on a alors $\mathbf{V}(\mathrm{I}') \subset \mathbb{A}^{n+1} \setminus \mathbf{V}(g) \subset \mathbb{A}^{n+1} \setminus \mathbf{V}(\mathrm{I})$. ■

Exercice 25.1 — Test d'appartenance au radical.

1. Montrer que g appartient au radical de I si, et seulement si, 1 appartient à I'.
2. En déduire un algorithme pour tester l'appartenance de g au radical de I.

■

Nullstellensatz fort

Tous les éléments sont maintenant réunis pour conclure la preuve du Nullstellensatz affine fort.

Preuve du Théorème 25.1. Avec les notations de la Proposition 25.5, $\mathbf{V}(\mathrm{I}') = \varnothing$ pour $\mathrm{I}' = \mathrm{I} + (\mathrm{Y}g - 1)$, si bien que I' contient 1 par le Théorème 25.4. Il existe donc des polynômes p_i et un polynôme p dans $k[\mathrm{X}_1,\dots,\mathrm{X}_n,\mathrm{Y}]$ tels que

$$1 = \sum_{i=1}^{s} p_i(\mathrm{X}_1,\dots,\mathrm{X}_n,\mathrm{Y}) f_i(\mathrm{X}_1,\dots,\mathrm{X}_n) + p(\mathrm{X}_1,\dots,\mathrm{X}_n,\mathrm{Y})(\mathrm{Y}g(\mathrm{X}_1,\dots,\mathrm{X}_n) - 1).$$

La substitution de $1/g(\mathrm{X}_1,\dots,\mathrm{X}_n)$ à Y dans cette égalité donne

$$1 = \sum_{i=1}^{s} p_i(\mathrm{X}_1,\dots,\mathrm{X}_n, 1/g(\mathrm{X}_1,\dots,\mathrm{X}_n)) f_i(\mathrm{X}_1,\dots,\mathrm{X}_n).$$

Algèbre		Géométrie
Idéal		Variété
$\mathbf{I}(V)$	$\longleftarrow$	V
I	$\longrightarrow$	$\mathbf{V}(I)$
$\sqrt{I}$	$\longleftarrow$	$\mathbf{V}(I)$.

FIGURE 25.1 – Liens entre algèbre et géométrie.

Multiplier les deux membres par g^e avec $e = \max_{1\leq i\leq s}(\deg_Y p_i)$ mène à

$$g(X_1,\dots,X_n)^e = \sum_{i=1}^{s} g(X_1,\dots,X_n)^e p_i(X_1,\dots,X_n, 1/g(X_1,\dots,X_n)) f_i(X_1,\dots,X_n)$$

avec $g(X_1,\dots,X_n)^e p_i(X_1,\dots,X_n, 1/g(X_1,\dots,X_n)) \in k[X_1,\dots,X_n]$. Ceci prouve que g^e est dans I. ■

Corollaire 25.6 Supposons k algébriquement clos, et soit I un idéal, alors on a $\mathbf{I}(\mathbf{V}(I)) = \sqrt{I}$.

Démonstration. Le Nullstellensatz affine fort donne $\mathbf{I}(\mathbf{V}(I)) \subseteq \sqrt{I}$. L'inclusion inverse est immédiate. ■

Un résumé des théorèmes précédents est donné par la Figure 25.1. Une conséquence importante de ces résultats est que pour prouver un théorème de géométrie, comme le théorème d'Apollonius présenté dans l'Exemple 23.1 au début de cette partie, il faut tester que les polynômes représentant la conclusion de l'énoncé appartiennent au radical de l'idéal engendré par les polynômes construits à partir des hypothèses.

25.2 Idéaux de dimension zéro

Une première application du Nullstellensatz est un lien entre l'escalier et la variété affine de certains idéaux.

Théorème 25.7 Soit I un idéal propre de $R = k[X_1,\dots,X_n]$. Les assertions suivantes sont équivalentes :

i. le complémentaire de l'escalier E(I) est fini pour tout ordre admissible ;
ii. il existe un ordre admissible pour lequel le complémentaire de E(I) est fini ;
iii. $\dim_k R/I$ est finie ;
iv. la variété affine $\mathbf{V}_{\bar{k}}(I)$ contient un nombre fini de points (où $\bar{k}$ représente la clôture algébrique de k).

Démonstration. Il est clair que (i) implique (ii). Comme $R/I = \bigoplus_{a\notin E(I)} kx^a$ (par le Théorème 23.14), la dimension $\dim_k R/I$ est le cardinal du complémentaire de l'escalier, et donc (ii) implique (iii).

Si (iii) est vrai alors notons $\delta = \dim_k \mathrm{R/I} < \infty$. Pour tout $j \in \{1,\dots,n\}$, la famille $1, \mathrm{X}_j, \dots, \mathrm{X}_j^{\delta}$ est liée dans R/I. Donc il existe $\mathrm{P}_j \in k[\mathrm{X}_j]$ qui appartient à l'idéal I. Par conséquent on a $\mathbf{V}_{\bar{k}}(\mathrm{I}) \subset \bigcap_{j=1}^{n} \mathbf{V}_{\bar{k}}(\mathrm{P}_j)$, ce qui implique (iv).

Si (iv) est vrai, alors pour tout $j \in \{1,\dots,n\}$, la projection de $\mathbf{V}_{\bar{k}}(\mathrm{I})$ sur l'axe porté par X_j est finie. Il existe donc $\mathrm{P}_j \in k[\mathrm{X}_j]$ qui s'annule sur cette projection. Par le Nullstellensatz affine fort (Théorème 25.1), on déduit de $\mathbf{V}_{\bar{k}}(\mathrm{I}) \subseteq \mathbf{V}_{\bar{k}}(\mathrm{P}_j)$ l'existence de $r_j \in \mathbb{N}$ tel que $\mathrm{P}_j^{r_j} \in \mathrm{I} \cap k[\mathrm{X}_j]$. Donc l'escalier touche chaque axe et son complémentaire est fini quel que soit l'ordre admissible considéré, ce qui implique (i). ■

Définition 25.2 Les idéaux vérifiant la proposition précédente sont appelés les *idéaux zéro-dimensionnels*, ou *de dimension 0*.

Polynômes caractéristique et minimal

Une conséquence du Théorème 25.7 est que certaines propriétés des idéaux de dimension 0 et de leurs variétés peuvent être calculées par des algorithmes d'algèbre linéaire. Un point de départ est fourni par l'énoncé suivant.

Lemme 25.8 Soit I un idéal de R de dimension 0. À tout polynôme $f \in \mathrm{R}$ on peut associer un endomorphisme de l'espace vectoriel $\mathrm{B} = \mathrm{R/I}$, qui en multiplie les éléments par f mod I, et que l'on note m_f.

Soient $\chi_f(\mathrm{T})$ et $\mu_f(\mathrm{T})$ les polynômes caractéristique et minimal de m_f. Alors, $\mu_f(f)$ et $\chi_f(f)$ appartiennent à I, et $\chi_f(\mathrm{T})$ et $\mu_f(\mathrm{T})$ s'annulent aux valeurs que prend f sur $\mathbf{V}_{\bar{k}}(\mathrm{I})$, et uniquement en ces valeurs. De plus, si l'idéal I est radical, alors $\mu_f(\mathrm{T})$ est la partie sans carré de $\chi_f(\mathrm{T})$.

Démonstration. L'idéal étant de dimension 0, il est différent de (1) et le monôme 1 appartient donc au quotient B. Son produit par f est nul dans B si, et seulement si, $f \in \mathrm{I}$, donc aussi si, et seulement si, m_f est nul. Si r est un polynôme de $k[\mathrm{T}]$, alors $m_{r(f)} = r(m_f)$ se vérifie en prouvant que multiplication par un scalaire, addition et multiplication sur les matrices produisent les applications souhaitées. Donc le polynôme minimal de la multiplication par f vérifie $\mu_f(m_f) = 0 = m_{\mu_f(f)}$, ce qui montre que $\mu_f(f)$ appartient à I et donc aussi son multiple $\chi_f(f)$. Ces deux polynômes s'annulent donc sur $\mathbf{V}_{\bar{k}}(\mathrm{I})$ et donc μ_f et χ_f s'annulent sur les valeurs que prend f sur $\mathbf{V}_{\bar{k}}(\mathrm{I})$.

Réciproquement, si α est une racine du polynôme caractéristique χ_f, alors c'est une valeur propre de m_f, ce qui signifie qu'il existe v non nul dans B tel que $(f-\alpha)v = 0$. Si α n'est pas une valeur de f sur un élément de $\mathbf{V}_{\bar{k}}(\mathrm{I})$, alors par le Nullstellensatz faible (Théorème 25.4), l'idéal engendré par I et le polynôme $f-\alpha$ contient 1, ou autrement dit $f-\alpha$ admet un inverse g modulo I. En multipliant $(f-\alpha)v$ par g, on obtient $v = 0$, une contradiction.

Si l'idéal I est radical, l'appartenance de $\chi(f)$ à I entraîne celle de $\bar{\chi}(f)$, où $\bar{\chi}$ est la partie sans carré de f. Par minimalité, le polynôme minimal est donc égal à $\bar{\chi}$. ■

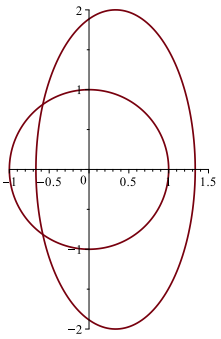


FIGURE 25.2 – Intersection d'un cercle et d'une ellipse.

Exemple 25.2 Les polynômes

$$f_1 = X_1^2 + X_2^2 - 1 \quad \text{et} \quad f_2 = 36X_1^2 + 9X_2^2 - 24X_1 - 32$$

décrivent l'intersection d'un cercle et d'une ellipse, représentés en Figure 25.2. Une base standard de l'idéal (f_1, f_2) dans $\mathbb{Q}[X_1, X_2]$ pour l'ordre `tdeg` est fournie par les deux polynômes

$$g_1 = f_2 - 9f_1 = 27X_1^2 - 24X_1 - 23, \quad g_2 = 36f_1 - f_2 = 27X_2^2 + 24X_1 - 4.$$

Ceci fournit d'abord une base du quotient $B = \mathbb{Q}[X_1, X_2]/(f_1, f_2)$ via les monômes sous l'escalier, à savoir $(1, X_1, X_2, X_1X_2)$. Ensuite, les matrices de multiplication par X_1 et par X_2 dans cette base valent

$$\begin{pmatrix} 0 & \frac{23}{27} & 0 & 0 \\ 1 & \frac{8}{9} & 0 & 0 \\ 0 & 0 & 0 & \frac{23}{27} \\ 0 & 0 & 1 & \frac{8}{9} \end{pmatrix}, \qquad \begin{pmatrix} 0 & 0 & \frac{4}{27} & -\frac{184}{243} \\ 0 & 0 & -\frac{8}{9} & -\frac{52}{81} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Leurs polynômes caractéristiques sont

$$\chi_{X_1}(T) = \frac{1}{729}(27T^2 - 24T - 23)^2, \quad \chi_{X_2}(T) = T^4 + \frac{40}{81}T^2 - \frac{560}{729}.$$

Le premier n'est pas minimal, puisque sa racine carrée g_1 est dans l'idéal. Les racines de ce polynôme, $(4 \pm \sqrt{85})/9$ sont les valeurs que prend le polynôme X_1 sur les points d'intersection (c'est-à-dire leurs abscisses sur la figure). Seule l'une de ces abscisses est visible sur la figure, les deux autres points d'intersection ayant une ordonnée qui n'est pas réelle. Le polynôme χ_{X_2} est quant à lui sans carré, donc aussi minimal. Ses quatre racines sont les valeurs des ordonnées des quatre points d'intersection dans $\mathbb{C}^2$. De plus, pour chacune de ces valeurs de X_2, on

peut retrouver l'abscisse correspondante grâce à une base standard pour l'ordre lexicographique induit par $X_1 > X_2$, formée de

$$\chi_{X_2}(X_2), \quad 24X_1 + 27X_2^2 - 4.$$

Paramétrisation par élément primitif

Des bases de l'espace vectoriel R/I commodes pour le calcul sont obtenues à partir d'éléments primitifs dans le sens suivant :

Définition 25.3 Si I est un idéal zéro-dimensionnel de R, un élément u de $B := R/I$ est dit *primitif* si ses puissances $1, u, \ldots, u^{\dim B - 1}$ forment une base de B.

Exemple 25.3 Dans l'exemple précédent, X_2 est un élément primitif de B, mais X_1 n'en est pas un.

Exemple 25.4 Avec $n = 2$, considérons l'idéal $I = (X_1, X_2)^2$, et soit u un élément de $k[X_1, X_2]$. En écrivant u sous la forme $u_0 + u_1(X_1, X_2)X_1 + u_2(X_1, X_2)X_2$ avec $u_0 \in k$, on observe que $(u - u_0)^2$ appartient à I. L'espace engendré par les puissances de u est donc de dimension 2. Comme $B = k[X_1, X_2]/I$ est de dimension 3, il s'ensuit que I n'admet pas d'élément primitif. Nous allons voir que les idéaux radicaux admettent des éléments primitifs. Il ne s'agit néanmoins pas d'une condition nécessaire car X_1 est primitif pour $I = (X_1^2) \subset k[X_1]$.

Proposition 25.9 Si I est un idéal zéro-dimensionnel de R qui admet un élément primitif u, alors il existe d'uniques polynômes $v_1, \ldots, v_n$ de $k[T]$ de degrés inférieurs stricts à $\delta = \dim B$ tels que

$$I = (\mu_u(u), X_1 - v_1(u), \ldots, X_n - v_n(u)).$$

Démonstration. Puisque u est primitif, les polynômes $1, u, \ldots, u^{\delta-1}$ sont linéairement indépendants sur k modulo I. Par conséquent le degré de μ_u est δ. Par ailleurs, pour $i \in \{1, \ldots, n\}$, les $\delta + 1$ polynômes $X_i, 1, u, \ldots, u^{\delta-1}$ sont linéairement dépendants sur k dans B alors que $1, u, \ldots, u^{\delta-1}$ ne le sont pas, et donc chaque variable X_i s'exprime de façon unique comme une k-combinaison linéaire de $1, u, \ldots, u^{\delta-1}$. ■

Définition 25.4 — Paramétrisation par élément primitif. L'élément primitif u, son polynôme minimal μ_u, et les polynômes $v_1, \ldots, v_n$ de la proposition précédente sont appelés une *paramétrisation par élément primitif* de I.

Cette terminologie provient du fait que chaque racine α de μ_u dans $\bar{k}$ donne la valeur que prend l'élément primitif sur un point de $\mathbf{V}_{\bar{k}}(I)$, les coordonnées de ce point étant alors $(v_1(\alpha), \ldots, v_n(\alpha))$, et tous les points sont obtenus ainsi.

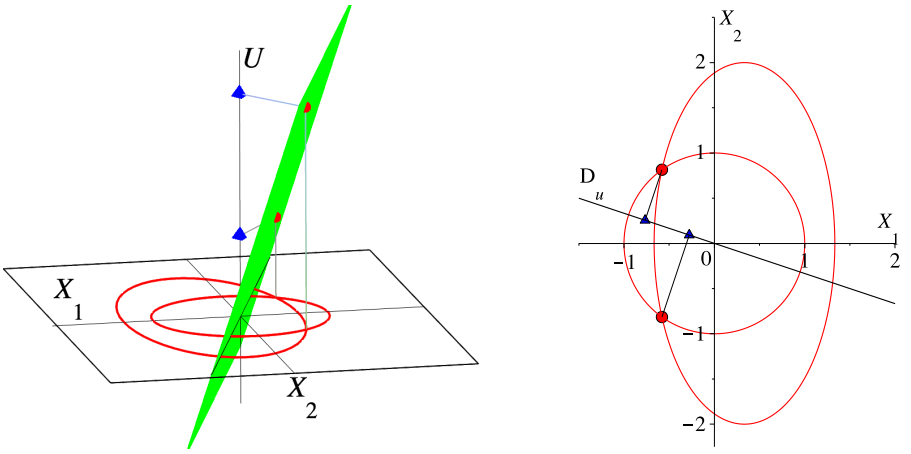


FIGURE 25.3 – Représentation par élément primitif de l'intersection d'une ellipse et d'un cercle (Exemple 25.5).

R Le cas favorable de l'Exemple 25.2 se généralise comme suit. Si I est un idéal radical zéro-dimensionnel de R et si X_n en est un élément primitif, alors une base standard de I pour l'ordre 1ex est de la forme

$$I = (\mu_{X_n}(X_n), X_1 - v_1(X_n), \ldots, X_{n-1} - v_{n-1}(X_n))$$

où $v_1(X_n), \ldots, v_{n-1}(X_n)$ sont des polynômes de $k[X_n]$. En effet, le polynôme $\mu_{X_n}(X_n)$ appartient à I d'après le Lemme 25.8 et donc à une base standard pour l'ordre 1ex. Le complémentaire de l'escalier E(I) est donc donné par les $\delta = \dim B$ monômes $1, X_n, \ldots, X_n^{\delta-1}$ et tout autre monôme X_i $(1 \leq i < n)$ se réduit à une combinaison linéaire de ces monômes.

Interprétation géométrique de la paramétrisation par élément primitif

Soit $u, \mu_u(T), v_1(T), \ldots, v_n(T)$ une paramétrisation par élément primitif d'un idéal zéro-dimensionnel I de R, avec u donné sous la forme $u = \lambda_1 X_1 + \cdots + \lambda_n X_n$ pour des λ_i dans k et

$$I = (\mu_u(u), X_1 - v_1(u), \ldots, X_n - v_n(u)).$$

Une interprétation géométrique commode est obtenue en ajoutant une nouvelle variable U, et en considérant l'idéal $\tilde{I} = I + (U - u)$ de $k[U, X_1, \ldots, X_n]$. On a alors

$$\tilde{I} = (\mu_u(U), U - u, X_1 - v_1(U), \ldots, X_n - v_n(U)).$$

La variété affine $\mathbf{V}_k(\tilde{I})$ de $\mathbb{A}^{n+1}$ est l'intersection de $k \times \mathbf{V}_k(I)$ par l'hyperplan d'équation $U = \lambda_1 X_1 + \cdots + \lambda_n X_n$. La projection π_U de $\mathbf{V}_k(\tilde{I})$ sur la coordonnée U est exactement $\mathbf{V}_k(\mu_u(U)) \subset \mathbb{A}^1$, c'est-à-dire l'ensemble des racines de μ_u dans k. En effet, chaque point de $\mathbf{V}_k(I)$ donne une valeur dans k à U puisque les λ_i sont dans k, et donc l'intersection avec l'hyperplan contient un point pour chaque élément de $\mathbf{V}_k(I)$, et réciproquement, pour chaque racine α de μ_u dans k l'ensemble $\pi_U^{(-1)}(\alpha)$ est réduit à un seul point, dont la projection sur les coordonnées $(X_1, \ldots, X_n)$ est un point de la variété $\mathbf{V}(I)$ initiale, qui sont ainsi tous obtenus.

Exemple 25.5 Avec l'idéal $I = (f_1, f_2)$ de l'Exemple 25.2, qui décrit l'intersection d'un cercle et d'une ellipse, la forme linéaire X_2 est un élément primitif de R/I. Il n'est pas utile d'augmenter la dimension pour donner une interprétation géométrique : la projection de l'intersection sur l'axe des X_2 est donnée par les zéros de χ_{X_2} et chacun d'entre eux correspond à un point dont l'autre coordonnée est donnée par la paramétrisation. Cette interprétation simple s'applique chaque fois que l'élément primitif utilisé pour la paramétrisation est une des coordonnées.

La construction ci-dessus peut être illustrée avec un autre choix d'élément primitif, par exemple $u = X_2 - 3X_1$. La Figure 25.3 montre (à gauche) le cercle et l'ellipse, l'intersection des droites verticales passant par leurs intersections avec le plan d'équation $U = X_2 - 3X_1$, et les projections de ces derniers points sur l'axe des U.

Lorsque le corps k est $\mathbb{R}$, on peut donner une autre interprétation géométrique, qui reste dans $\mathbb{A}^n$, dont on utilise alors la structure euclidienne. Les points de $\mathbf{V}(I)$ sont projetés orthogonalement sur la droite D_u passant par l'origine et orthogonale à l'hyperplan d'équation $\lambda_1 X_1 + \cdots + \lambda_n X_n = 0$. Autrement dit, la projection effectuée est

$$(X_1, \ldots, X_n) \mapsto \frac{\lambda_1 X_1 + \cdots + \lambda_n X_n}{\lambda_1^2 + \cdots + \lambda_n^2} (\lambda_1, \ldots \lambda_n).$$

Les coordonnées sur l'axe des U de la construction précédente deviennent des coordonnées sur la droite D_u et le reste de l'interprétation est identique. La partie droite de la Figure 25.3 illustre cette construction sur l'Exemple 25.5.

Généricité des éléments primitifs pour les idéaux radicaux

Il n'est pas difficile de trouver des éléments primitifs. Nous commençons par introduire une contrainte technique sur le corps k, afin d'éviter la situation de l'exemple suivant.

Exemple 25.6 Considérons l'idéal $I = (X_1^2 + Z) \subset \mathbb{F}_2(Z)[X_1]$, où $\mathbb{F}_2$ est le corps fini à deux éléments. Il est radical, et X_1 en est clairement un élément primitif. En revanche le polynôme minimal $\mu_{X_1}(T) = T^2 + Z$ de X_1 n'est pas séparable (Définition 18.1). En particulier, sa dérivée est 0, qui n'est pas inversible modulo μ_{X_1}.

Par le Lemme 25.8, lorsque I est radical, les polynômes minimaux μ_u sont sans carré. Une condition suffisante pour garantir qu'ils sont de plus séparables est de supposer que la caractéristique du corps de base k est nulle ou bien strictement plus grande que $\deg \mu_u$. Il s'agit d'un résultat classique, vu dans le Théorème 18.10 : les factorisations sans carré et séparable coïncident en caractéristique nulle ou bien strictement supérieure au degré du polynôme. D'après la Proposition 18.5, la dérivée μ_u' est alors inversible modulo μ_u. La proposition suivante caractérise l'ensemble des formes linéaires qui ne sont pas primitives.

Proposition 25.10 Soit I un idéal radical de dimension 0 de R, et supposons k de caractéristique nulle ou bien $> \delta = \dim \mathrm{R/I}$. Alors il existe une variété algébrique propre W de $\mathbb{A}^n$ telle que pour tout $(\lambda_1, \ldots, \lambda_n)$ à l'extérieur de W, la forme linéaire $u = \lambda_1 \mathrm{X}_1 + \cdots + \lambda_n \mathrm{X}_n$ est primitive pour I.

La preuve de cette proposition passe par la construction d'un polynôme non nul auquel la variété W est associée. Cette construction est algébrique et commence par introduire un n-uplet de nouvelles variables $\Lambda_1, \ldots, \Lambda_n$, et les corps et anneaux

$$k_\Lambda = k(\Lambda_1, \ldots, \Lambda_n), \quad \mathrm{R}_\Lambda = k_\Lambda[\mathrm{X}_1, \ldots, \mathrm{X}_n], \quad \mathrm{B}_\Lambda = \mathrm{R}_\Lambda/\mathrm{I}_\Lambda,$$

où I_Λ représente l'*extension* de I dans R_Λ, c'est-à-dire l'idéal engendré par les éléments de I dans cette anneau, qui se comporte de façon très similaire à I en vertu de la propriété suivante.

Lemme 25.11 Un idéal I de R et son extension I_Λ à R_Λ ont mêmes bases standard. L'extension I_Λ est donc zéro-dimensionnelle lorsque I l'est.

Démonstration. Une base standard de I engendre bien I_Λ dans R_Λ. De plus, ses S-polynômes se réduisent à zéro par la même réduction que dans R, ce qui en fait une base standard de I_Λ par le Théorème 24.3. Ayant mêmes bases standard, ces idéaux ont les mêmes escaliers, et donc sont zéro-dimensionnels simultanément. ■

Lemme 25.12 Soit $\mu_{u_\Lambda} \in k_\Lambda(\mathrm{T})$ le polynôme minimal de la multiplication par $u_\Lambda = \Lambda_1 \mathrm{X}_1 + \cdots + \Lambda_n \mathrm{X}_n$ dans B_Λ. Alors pour tout $i \in \{1, \ldots, n\}$, on a

$$\frac{\partial \mu_{u_\Lambda}}{\partial \mathrm{T}}(u_\Lambda)\mathrm{X}_i + \frac{\partial \mu_{u_\Lambda}}{\partial \Lambda_i}(u_\Lambda) \in \mathrm{I}_\Lambda. \tag{25.2}$$

Démonstration. Par noetherianité, on peut supposer l'idéal I engendré par des polynômes $f_1, \ldots, f_s$ de R. Par le Lemme 25.8, $\mu_{u_\Lambda}(u_\Lambda)$ appartient à I_Λ, c'est-à-dire qu'il existe des polynômes $g_1, \ldots, g_s$ dans l'anneau R_Λ tels que $\mu_{u_\Lambda}(u_\Lambda) = g_1 f_1 + \cdots + g_s f_s$, identité qu'il suffit de dériver par rapport à Λ_i pour obtenir le résultat. ■

Lorsque le polynôme $\partial \mu_{u_\Lambda}/\partial \mathrm{T}$ est inversible modulo $\mu_{u_\Lambda}(\mathrm{T})$, ce lemme donne une paramétrisation de $\mathbf{V}_{\bar{k}_\Lambda}(\mathrm{I})$ dont il s'agit maintenant de contrôler les spécialisations des Λ_i.

Exemple 25.7 Avec l'idéal $\mathrm{I} = (f_1, f_2)$ de l'Exemple 25.2, le polynôme caractéristique de la multiplication par $\Lambda_1 \mathrm{X}_1 + \Lambda_2 \mathrm{X}_2$ vaut

$$\chi_{u_\Lambda}(\mathrm{T}) = \mathrm{T}^4 - \frac{16}{9}\Lambda_1 \mathrm{T}^3 - \frac{1}{81}(74\Lambda_1^2 - 40\Lambda_2^2)\mathrm{T}^2 + \frac{16}{243}(23\Lambda_1^2 + 50\Lambda_2^2)\Lambda_1 \mathrm{T} + \frac{1}{729}(529\Lambda_1^4 - 760\Lambda_1^2\Lambda_2^2 - 560\Lambda_2^4).$$

Il est sans carré, c'est donc aussi le polynôme minimal $\mu_{u_\Lambda}(T)$. En $\Lambda_1 = 1, \Lambda_2 = 0$ (resp. en $\Lambda_1 = 0, \Lambda_2 = 1$), on retrouve le polynôme χ_{X_1} (reps. χ_{X_2}) de l'Exemple 25.2. Ensuite, la dérivée de μ_{u_Λ} par rapport à Λ_1, par exemple, vaut $-16T^3/9+800T/243$ en $\Lambda_1 = 0$, $\Lambda_2 = 1$. En multipliant ce polynôme par l'inverse de $\chi'_{X_2}(T)$ modulo $\chi_{X_2}(T)$, on obtient $9T^2/8 - 1/6$, c'est-à-dire qu'on retrouve le dernier polynôme de l'Exemple 25.2.

Il n'est pas acquis *a priori* que l'évaluation du polynôme minimal μ_{u_Λ} en des valeurs données λ_i des Λ_i donne le polynôme minimal de la multiplication par la forme linéaire spécialisée $u = \lambda_1 X_1 + \cdots + \lambda_n X_n$. C'est en revanche le cas pour le polynôme caractéristique.

Lemme 25.13 Soient χ_{u_Λ} et μ_{u_Λ} les polynômes caractéristique et minimal de la multiplication par $u_\Lambda = \Lambda_1 X_1 + \cdots + \Lambda_n X_n$ dans B_Λ. Alors χ_{u_Λ} et μ_{u_Λ} appartiennent à $k[\Lambda_1,\ldots,\Lambda_n][T]$, et pour tout $(\lambda_1,\ldots,\lambda_n) \in k^n$, le polynôme caractéristique de la multiplication par $u = \lambda_1 X_1 + \cdots + \lambda_n X_n$ dans B est égal à l'évaluation de χ_{u_Λ} en $\Lambda_1 = \lambda_1, \ldots, \Lambda_n = \lambda_n$.

Démonstration. On choisit une base de B formée de monômes sous l'escalier d'une base standard de I. Cette base est aussi base de B_Λ par le Lemme 25.11, et la matrice M_{u_Λ} de multiplication par u_Λ dans cette base a donc ses coefficients dans $k[\Lambda_1,\ldots,\Lambda_n]$. Il s'ensuit que χ_{u_Λ} et μ_{u_Λ} appartiennent tout deux à $k[\Lambda_1,\ldots,\Lambda_n][T]$. La matrice M_{u_Λ} se spécialise donc en la matrice M de multiplication par u lorsque qu'on remplace les Λ_i par les λ_i. Par conséquent, les polynômes caractéristiques de M_{u_Λ} et M sont bien spécialisations l'un de l'autre. ■

Dans le cas où I est radical, la situation se simplifie. D'abord, le fait d'être radical résiste à l'extension.

Lemme 25.14 Soit I un idéal radical de R. Alors son extension I_Λ à R_Λ est aussi un idéal radical.

Démonstration. Soit $f \in R_\Lambda$ tel que $f^e \in I_\Lambda$ pour un entier $e \geq 2$. Il existe un polynôme $a \in k[\Lambda_1,\ldots,\Lambda_n]$ tel que $g := af$ appartienne à $k[\Lambda_1,\ldots,\Lambda_n,X_1,\ldots,X_n]$, et il suffit de prouver que g appartient à I_Λ. D'abord, g^e étant un polynôme de $I_\Lambda \cap k[\Lambda_1,\ldots,\Lambda_n,X_1,\ldots,X_n]$, sa réduction par une base standard de I permet de déduire qu'il s'écrit comme une combinaison linéaire à coefficients polynomiaux de générateurs I.

Soit $<$ un ordre monomial sur les monômes en les seules variables $\Lambda_1,\ldots,\Lambda_n$. Si g est non nul, alors il se récrit sous la forme $g = \mathrm{ct}_<(g) + (g - \mathrm{ct}_<(g))$ et en extrayant les coefficients des monômes appropriés en les Λ_i dans la décomposition de g^e ci-dessus, on vérifie que $\mathrm{ct}_<(g)^e$ est dans I, ce qui conduit à $\mathrm{ct}(g) \in I$, puis à $(g - \mathrm{ct}_<(g))^e \in I_\Lambda$. En itérant cet argument, il s'ensuit que tous les coefficients de g dans R se trouvent dans I et donc que g est dans I_Λ. ■

Proposition 25.15 Soit I un idéal radical de R de dimension 0, et supposons k de caractéristique nulle ou $> \delta = \dim R/I$. Alors, χ_{u_Λ} est sans carré, égal à μ_{u_Λ}, et les équations (25.2) fournissent une paramétrisation par élément primitif de I_Λ.

Démonstration. D'après le lemme précédent, I_Λ est radical ce qui entraîne, d'après le Lemme 25.8, que le polynôme μ_{u_Λ} est sans carré. Grâce à l'hypothèse sur k, on obtient que μ_{u_Λ} est premier avec sa dérivée et les équations (25.2) se récrivent $X_i - v_{\Lambda,i}(u_\Lambda) \in I_\Lambda$, avec $v_i = -\frac{\partial \mu_{u_\Lambda}}{\partial \Lambda_i} / \frac{\partial \mu_{u_\Lambda}}{\partial T} \bmod \mu_{u_\Lambda}$. Ceci montre que tout monôme en les X_i se récrit comme combinaison linéaire de $1, u_\Lambda, \dots, u_\Lambda^{\deg \mu_{u_\Lambda} - 1}$, ce qui donne donc une borne sur la dimension de l'espace vectoriel B_Λ et donc sur le degré de χ_{u_Λ}. Par conséquent, χ_{u_Λ} coïncide avec son diviseur μ_{u_Λ}. ■

Corollaire 25.16 Soit I un idéal radical de R de dimension 0, et supposons k de caractéristique nulle ou $> \delta = \dim R/I$. Soit $(\lambda_1, \dots, \lambda_n)$ un point de k^n qui n'annule pas le discriminant de $\mu_{u_\Lambda}(T)$, et posons $u = \lambda_1 X_1 + \cdots + \lambda_n X_n$. Alors, le polynôme minimal $\mu_u(T)$ est la spécialisation de χ_{u_Λ} en $\Lambda_1 = \lambda_1, \dots, \Lambda_n = \lambda_n$, et cette spécialisation, appliquée sur les équations (25.2), induit une paramétrisation de I par l'élément primitif u.

Démonstration. D'après la proposition précédente, $\chi_{u_\Lambda} = \mu_{u_\Lambda}$ est sans carré. Cette propriété est donc préservée par l'évaluation des Λ_i en les λ_i qui n'annulent pas son discriminant. Le Lemme 25.13 montre que le polynôme ainsi obtenu est le polynôme caractéristique χ_u de la multiplication par u, qui est donc son polynôme minimal μ_u, puisqu'il est sans carré. L'hypothèse sur k entraine que $\mu_u'(T)$ et $\mu_u(T)$ sont premiers entre eux, et donc les polynômes de l'Équation (25.2) se spécialisent puis se récrivent sous la forme $X_i - v_i(u) \in I$ après inversion de μ_u' modulo μ_u. Ceci montre que $1, u, \dots, u^{\deg \mu_u - 1}$ engendrent bien R/I. ■

Fin de la démonstration de la Proposition 25.10. L'ensemble des zéros du discriminant de μ_{u_Λ} fournit donc une variété W ayant la propriété requise. ■

Exercice 25.2 Soit I un idéal radical de dimension 0. En supposant k de caractéristique nulle ou strictement plus grande que $\dim R/I$, montrer qu'un polynôme u est primitif pour I si, et seulement, s'il prend des valeurs distinctes en des points distincts de $\mathbf{V}_{\bar{k}}(I)$. ■

La construction de paramétrisations que nous venons de voir, motive naturellement la définition suivante.

Définition 25.5 — Paramétrisation sous forme de Kronecker. Une paramétrisation d'un idéal I de dimension 0 sous *forme de Kronecker* est la donnée d'un élément primitif u, de son polynôme minimal μ_u, et de polynômes $w_1, \dots, w_n$ tels que

$$I = (\mu_u(u), \mu_u'(u)X_1 - w_1(u), \dots, \mu_u'(u)X_n - w_n(u)).$$

Une telle paramétrisation est bien entendu équivalente à celle de la Définition 25.5, dans la mesure où on peut aisément passer de l'une à l'autre par les formules suivantes :

$$v_i = (\mu_u')^{-1} w_i \bmod \mu_u, \quad w_i = \mu_u' v_i \bmod \mu_u.$$

Néanmoins, en général, quand vient la question pratique de la taille des coefficients de ces paramétrisations, il est intéressant de constater que la forme de Kronecker est plus compacte. D'abord, compte-tenu de la construction des w_i que nous venons de voir, il est naturel de s'attendre à ce que les tailles des coefficients des w_i et de μ_u soient du même ordre de grandeur. Ensuite, inverser μ_u' modulo μ fait en général croître significativement la taille des coefficients. Nous ne formaliserons pas ces assertions ici, mais nous contenterons de les appuyer sur l'exemple suivant :

Exemple 25.8 Avec $k = \mathbb{Q}$, $n = 2$, on prend pour f_1 et f_2 des polynômes de degrés partiels 3 à la fois en X_1 et X_2, avec des coefficients pris au hasard entre -99 et 99. Un calcul de base standard pour l'ordre 1ex nous donne une paramétrisation pour $u = X_2$. Les coefficients de μ_u sont des fractions rationnelles composées d'entiers d'environ 15 chiffres. Les coefficients de v_1 font intervenir des entiers de plus de 130 chiffres. Ceux de w_1 ont moins de 30 chiffres. Ici le degré de μ_u est 18.

Dans le chapitre suivant nous étudierons un cas très spécial de paramétrisation, lorsque k est un corps de fractions rationnelles. Nous verrons que les degrés totaux des polynômes d'une paramétrisation sous forme de Kronecker sont bornés par la même quantité. Cette borne est au cœur de l'analyse de complexité de l'algorithme de *résolution géométrique* du Chapitre 28.

25.3 Projection des variétés affines

Nous allons voir maintenant que le pendant géométrique de l'élimination de la variable X_n correspond à la projection

$$\begin{array}{rccc} \pi : & \mathbb{A}^n & \to & \mathbb{A}^{n-1} \\ & (x_1,\ldots,x_n) & \mapsto & (x_1,\ldots,x_{n-1}). \end{array}$$

Théorème 25.17 Soit I un idéal de R. La projection $\pi(\mathbf{V}(I))$ de la variété affine $\mathbf{V}(I)$ est contenue dans la variété affine $\mathbf{V}(I \cap k[X_1,\ldots,X_{n-1}])$.

De plus, si k est algébriquement clos, alors $\mathbf{V}(I \cap k[X_1,\ldots,X_{n-1}])$ est la plus petite variété contenant $\pi(\mathbf{V}(I))$. On dit que c'est la *clôture de Zariski* de $\pi(\mathbf{V}(I))$.

Démonstration. Soient $x = (x_1,\ldots,x_{n-1}) \in \pi(\mathbf{V}(I))$ et $f \in I \cap k[X_1,\ldots,X_{n-1}]$. Il existe $x_n \in k$ tel que $x' = (x_1,\ldots,x_n) \in \mathbf{V}(I)$ donc $f(x') = 0$. Comme on peut aussi voir f comme un polynôme de $k[X_1,\ldots,X_{n-1}]$, on peut écrire $f(x) = 0$. Par conséquent, $\pi(\mathbf{V}(I)) \subseteq \mathbf{V}(I \cap k[X_1,\ldots,X_{n-1}])$.

Soit $f \in k[X_1,\ldots,X_{n-1}]$ s'annulant identiquement sur tout $\pi(\mathbf{V}(I)) \subseteq \mathbb{A}^{n-1}$. Puisque f ne dépend pas de X_n alors f, vu comme un polynôme de R, s'annule identiquement

sur $\pi(\mathbf{V}(\mathrm{I}))\times k \subseteq \mathbb{A}^n$. Comme $\mathbf{V}(\mathrm{I}) \subseteq \pi(\mathbf{V}(\mathrm{I}))\times k$, le Nullstellensatz fort (Théorème 25.1) nous donne $f \in \sqrt{\mathrm{I}}$. Finalement f s'annule bien sur $\mathbf{V}(\mathrm{I}\cap k[\mathrm{X}_1,\dots,\mathrm{X}_{n-1}])$, ce qui prouve que cette dernière est bien la plus petite variété contenant $\pi(\mathbf{V}(\mathrm{I}))$. ■

Exemple 25.9 Soit la variété d'équation $\mathrm{X}_1\mathrm{X}_2 - 1 = 0$. Alors la projection de l'hyperbole sur l'axe des X_1 est toute la droite privée de l'origine. La plus petite variété la contenant est bien $\mathbf{V}((0))$ car $(\mathrm{X}_1\mathrm{X}_2 - 1)\cap k[\mathrm{X}_1] = (0)$. On voit qu'il peut manquer des points lorsque l'on projette des variétés affines.

On peut préciser le théorème précédent en étudiant le lieu des points qui n'appartiennent pas à la projection.

Proposition 25.18 Supposons k algébriquement clos, et soit $\mathrm{I} = (f_1,\dots,f_s)$ et $\mathrm{I}' = \mathrm{I}\cap k[\mathrm{X}_1,\dots,\mathrm{X}_{n-1}]$ son premier idéal éliminant. Considérons les polynômes f_i comme des polynômes en X_n à coefficients dans $k[\mathrm{X}_1,\dots,\mathrm{X}_{n-1}]$ et notons $g_i \in k[\mathrm{X}_1,\dots,\mathrm{X}_{n-1}]$ le terme de tête de f_i. Alors

$$\mathbf{V}(\mathrm{I}') = \pi(\mathbf{V}(\mathrm{I})) \cup (\mathbf{V}(g_1,\dots,g_s)\cap \mathbf{V}(\mathrm{I}')).$$

En d'autres termes, tout point de $\mathbf{V}(\mathrm{I}')$ qui n'annule pas tous les termes de tête est la projection d'un point de $\mathbf{V}(\mathrm{I})$.

Démonstration. Il suffit de montrer que $\mathbf{V}(\mathrm{I}') \setminus \mathbf{V}(g_1,\dots,g_s) \subseteq \pi(\mathbf{V}(\mathrm{I}))$. Soit un point $a = (a_1,\dots,a_{n-1}) \in \mathbf{V}(\mathrm{I}')$. Sans perte de généralité on peut supposer que $g_1(a) \neq 0$. Soit $f \in \mathrm{I}$ et considérons $g := \mathrm{Res}_{\mathrm{X}_n}(f_1, f)$. Par la Proposition 6.7 on a $g \in \mathrm{I}'$ et donc $g(a_1,\dots,a_{n-1}) = 0$. Par ailleurs, comme $g_1(a_1,\dots,a_{n-1}) \neq 0$, on peut rendre f_1 unitaire en X_n en le voyant comme un polynôme à coefficients dans les séries formelles $k[[\mathrm{X}_1 - a_1,\dots,\mathrm{X}_{n-1} - a_{n-1}]]$. La Proposition 6.8 montre alors que $g(a_1,\dots,a_{n-1})$ est proportionnel sur k à

$$\mathrm{Res}_{\mathrm{X}_n}(f_1(a_1,\dots,a_{n-1},\mathrm{X}_n), f(a_1,\dots,a_{n-1},\mathrm{X}_n)),$$

ce qui implique que $f(a_1,\dots,a_{n-1},\mathrm{X}_n)$ n'est pas constant. Finalement l'idéal

$$\mathrm{H} := \{f(a_1,\dots,a_{n-1},\mathrm{X}_n) \mid f \in \mathrm{I}\}$$

de $k[\mathrm{X}_n]$ est propre, ce qui nous permet de conclure que $\pi^{-1}(a)\cap\mathbf{V}(\mathrm{I})$ est non vide. ■

Corollaire 25.19 S'il existe un polynôme unitaire en X_n dans I, alors $\mathbf{V}(\mathrm{I}') = \pi(\mathbf{V}(\mathrm{I}))$.

Exemple 25.10 Dans l'exemple précédent, l'hyperbole décrite par $\mathrm{X}_1\mathrm{X}_2 - 1 = 0$ est projetée sur l'axe des X_1 et seul le point 0 de l'axe des X_1 n'est pas un projeté. Après le changement de variables $\mathrm{X}_1 \leftarrow \mathrm{X}_1 + \mathrm{X}_2$, f devient $\mathrm{X}_2^2 + \mathrm{X}_1\mathrm{X}_2 - 1$, et on se trouve dans la situation du corollaire, où la projection est bien surjective. Cette

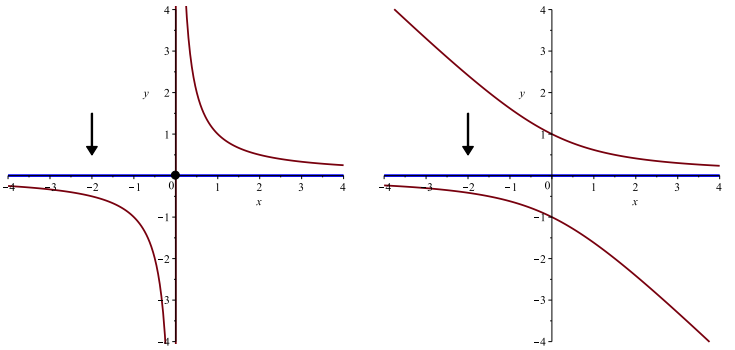


FIGURE 25.4 – Les projections de l'hyperbole.

situation est illustrée dans la Figure 25.4.

25.4 Nullstellensatz projectif

Dans cette section et la suivante, nous revenons au cas projectif, et R représente à partir de maintenant l'anneau $k[X_0,\dots,X_n]$.

Définition 25.6 Le cône affine $\widetilde{V}$ d'une variété projective V est l'ensemble des représentants des éléments de V, augmenté de 0 :

$$\widetilde{V} := \{(x_0,\dots,x_n) \in \mathbb{A}^{n+1} \mid (x_0,\dots,x_n) = (0,\dots,0) \text{ ou } (x_0 : \cdots : x_n) \in V\}.$$

Ainsi, si I est un idéal homogène de $k[X_0,\dots,X_n]$ alors le cône affine de la variété projective de I est égal à la variété affine de I.

Théorème 25.20 — Nullstellensatz projectif faible. Si k est algébriquement clos, si I est un idéal homogène de R tel que $\mathbf{V}(I) = \varnothing$, alors il existe $r \in \mathbb{N}$ tel que $J^r \subseteq I$, où J est l'idéal $(X_0,\dots,X_n)$ des polynômes s'annulant à l'origine.

Démonstration. Le cône affine $\widetilde{\mathbf{V}(I)}$ de $\mathbf{V}(I)$ est réduit au point $(0,\dots,0)$. Pour tout $i \in \{1,\dots,n\}$, le polynôme X_i s'annule identiquement sur $\widetilde{\mathbf{V}(I)}$. Par le Nullstellensatz affine fort (Théorème 25.1), il existe $r_i \in \mathbb{N}$ tel que $X_i^{r_i} \in I$. Soient $m = \max_{1\le i\le n}(r_i)$ et $r = (n+1)(m-1)+1$. Alors, l'idéal J^r est l'idéal engendré par les monômes de degré r. Si X^α est un tel monôme avec $\alpha = (\alpha_0,\dots,\alpha_n) \in \mathbb{N}^{n+1}$ et $|\alpha| = \sum_i \alpha_i = r$, alors il existe i tel que $\alpha_i \ge m$ donc $X^\alpha = X_i^r X^\beta \in I$ pour un certain $\beta \in \mathbb{N}^{n+1}$. Ainsi chaque générateur de J^r est dans I et $J^r \subseteq I$. ■

Théorème 25.21 — Nullstellensatz projectif fort. Si k est algébriquement clos et si I est un idéal homogène de R, alors tout polynôme $g \in R$ homogène qui s'annule

identiquement sur $\mathbf{V}(I)$ appartient à $\sqrt{I}$.

Démonstration. Comme dans la démonstration précédente, nous nous ramenons au cas affine en considérant le cône affine $\widetilde{\mathbf{V}(I)}$ dans $\mathbb{A}^{n+1}$ défini par I. Le polynôme g s'annule alors identiquement sur $\widetilde{\mathbf{V}(I)}$, ce qui implique par le Nullstellensatz fort que $g \in \sqrt{I}$. ■

Corollaire 25.22 Si k est algébriquement clos et si I est un idéal homogène, alors $\mathbf{I}(\mathbf{V}(I)) = \sqrt{I}$. En conséquence, $\sqrt{I}$ est un idéal homogène.

25.5 Projection des variétés projectives

Avant de pouvoir définir une projection dans un contexte projectif, commençons par quelques définitions et propriétés élémentaires sur les espaces linéaires projectifs.

Définition 25.7 Un *espace linéaire projectif* E est un sous-ensemble de points $(x_0 : \cdots : x_n)$ de l'espace projectif $\mathbb{P}^n$ vérifiant un système d'équations linéaires homogènes

$$\begin{cases} a_{1,0}x_0 & + & \cdots & + & a_{1,n}x_n & = & 0, \\ \vdots & & & & \vdots & & \vdots \\ a_{s,0}x_0 & + & \cdots & + & a_{s,n}x_n & = & 0. \end{cases}$$

Un *hyperplan* projectif est un espace linéaire projectif donné par une seule équation non triviale.

Autrement dit, un espace linéaire projectif F est l'image dans le quotient $\mathbb{P}^n$ d'un espace vectoriel $\widetilde{F}$, privé de 0. Alors $\widetilde{F}$ est le cône affine de F.

Définition 25.8 La somme $E + F$ de deux sous-espaces linéaires projectifs de $\mathbb{P}^n$ est l'image dans $\mathbb{P}^n$ de la somme de leurs cônes affines.

La notion de dimension s'étend naturellement aux espaces linéaires projectifs.

Définition 25.9 La dimension de l'espace linéaire projectif E, noté $\dim E$, est $\dim \widetilde{E} - 1$.

On vérifie facilement que $\mathbb{P}^n$ est de dimension n, que les hyperplans sont de dimension $n-1$ et ainsi de suite. Une propriété voulue du plan projectif est que deux droites se coupent toujours. Ceci est une conséquence de la proposition suivante en dimension quelconque.

Proposition 25.23 Soient E et F deux sous-espaces linéaires de $\mathbb{P}^n$. Alors

$$\dim(E+F)+\dim(E\cap F)=\dim E+\dim F.$$

Démonstration. Les espaces vectoriels associés $\tilde{E}$ et $\tilde{F}$ sont respectivement de dimension $\dim E+1$ et $\dim F+1$. Ainsi on a $\dim(\tilde{E}\cap\tilde{F})+\dim(\tilde{E}+\tilde{F})=\dim\tilde{E}+\dim\tilde{F}$. Comme $\tilde{E}\cap\tilde{F}=\widetilde{E\cap F}$ et $\tilde{E}+\tilde{F}=\widetilde{E+F}$, on déduit $\dim(E+F)+\dim(E\cap F)=\dim E+\dim F$. ■

Corollaire 25.24 Soient E et F deux sous-espaces linéaires de $\mathbb{P}^n$ tels que $\dim E+\dim F\geq n$. Alors $E\cap F\neq\varnothing$.

Démonstration. Par la proposition précédente, $\dim E\cap F\geq 0$. Donc $E\cap F$ n'est pas vide. ■

Nous allons définir les projections de l'espace projectif et vérifier que cela généralise la notion affine habituelle.

Définition 25.10 Soient L et B deux sous-espaces linéaires propres de $\mathbb{P}^n$ qui ne se coupent pas et tels que $\dim L+\dim B=n-1$. Alors la projection de *base* B avec L comme *centre de projection* est l'application

$$\pi:\left\{\begin{array}{ccc}\mathbb{P}^n\setminus L & \to & \mathbb{P}^n\\ x & \mapsto & (L+x)\cap B.\end{array}\right.$$

Le fait que l'application de projection π soit bien définie découle du Corollaire 25.24. Remarquons que cette définition généralise bien les projections affines usuelles. En effet soit la projection affine

$$\pi_n:\left\{\begin{array}{ccc}\mathbb{A}^n & \to & \mathbb{A}^{n-1}\\ (x_1,\dots,x_n) & \mapsto & (x_1,\dots,x_{n-1}).\end{array}\right.$$

Considérons l'espace affine $\mathbb{A}^n$ comme une partie de $\mathbb{P}^n$ via l'inclusion

$$\left\{\begin{array}{ccc}\mathbb{A}^n & \hookrightarrow & \mathbb{P}^n\\ (x_1,\dots,x_n) & \mapsto & (1:x_1:\cdots:x_n).\end{array}\right.$$

On vérifie que la projection affine π_n est la restriction à $\mathbb{A}^n$ de la projection sur la base $B=\mathbf{V}(X_n)$ et de centre à l'infini $L=(0:\cdots:0:1)$: en effet $(L+(1:x_1:\cdots:x_n))\cap B$ n'est autre que $(1:x_1:\cdots:x_{n-1}:0)$.

Contrairement au cas affine, le résultat suivant assure que les variétés projectives sont stables par projection.

Proposition 25.25 Supposons k algébriquement clos, et soient L et B deux sous-espaces linéaires propres de $\mathbb{P}^n$ qui ne se coupent pas et tels que $\dim L+\dim B=n-1$. Le projeté d'une variété projective disjointe de L par la projection de base B et de

centre de projection L est encore une variété projective.

Démonstration. Supposons que B soit de dimension r engendré par les points $l_0,\dots,l_r$ de $\mathbb{P}^n$, et que L soit engendré par $l_{r+1},\dots,l_n$. Pour tout $i \in \{r,\dots,n-1\}$, on pose $\mathrm{B}_i := \langle l_0,\dots,l_i\rangle$ et $\mathrm{L}_i := \langle l_{i+1},\dots,l_n\rangle$. Comme $\dim \mathrm{L}_i = n-i-1$ et $\dim \mathrm{B}_i = i$ et que B_i et L_i ne se coupent pas alors on peut considérer la projection π_i de base B_i et de centre L_i.

π_{n-1} est la projection sur l'hyperplan B_{n-1} de centre l_n. La restriction de π_{n-2} à B_{n-1} est la projection de base B_{n-2} et de centre l_{n-1}. Plus généralement, la restriction de π_{n-i} à B_{n-i+1} est la projection de base B_{n-i} et de centre l_{n-i+1}. Comme $\pi_r = \pi_r \circ \cdots \circ \pi_{n-1}$, il suffit de prouver le théorème lorsque $r = n-1$, ce que nous supposons désormais.

Sans perte de généralité nous pouvons effectuer un changement de variables linéaires pour nous ramener au cas où $\mathrm{B} = \mathbf{V}(\mathrm{X}_n)$ et $\mathrm{L} = (0 : \cdots : 0 : 1)$. On note π la projection correspondante. Comme L n'est pas dans $\mathbf{V}(\mathrm{I})$, il existe un polynôme homogène $f_1 \in \mathrm{I}$ ne s'annulant pas sur L. On pose $\mathrm{I}' = \mathrm{I} \cap k[\mathrm{X}_0,\dots,\mathrm{X}_{n-1}]$ et on considère un point $(a_0 : \cdots : a_{n-1}) \in \mathbf{V}(\mathrm{I}')$.

Soit $f \in \mathrm{I}$ et considérons $g := \mathrm{Res}_{\mathrm{X}_n}(f_1, f)$. Par la Proposition 6.7 on a $g \in \mathrm{I}'$ et donc $g(a_0,\dots,a_{n-1}) = 0$. Par ailleurs, comme $f_1(\mathrm{L}) \neq 0$, le degré partiel de f_1 en X_n coïncide avec son degré total et la Proposition 6.8 nous donne alors que $g(a_0,\dots,a_{n-1})$ est proportionnel sur k à

$$\mathrm{Res}_{\mathrm{X}_n}(f_1(a_0,\dots,a_{n-1},\mathrm{X}_n), f(a_0,\dots,a_{n-1},\mathrm{X}_n)),$$

ce qui implique que $f(a_0,\dots,a_{n-1},\mathrm{X}_n)$ n'est pas constant. Finalement l'idéal

$$\mathrm{H} := \{f(a_0,\dots,a_{n-1},\mathrm{X}_n) \mid f \in \mathrm{I}\}$$

de $k[\mathrm{X}_n]$ est propre, ce qui nous permet de conclure que $\pi^{-1}(a) \cap \mathbf{V}(\mathrm{I})$ est non vide. ■

Notes

Notre preuve du Théorème 25.4 est adaptée de l'article d'Arrondo [Arr06]. La forme particulière de la base standard décrite dans la remarque qui suit la Définition 25.4 a été appelée *shape lemma* par Gianni et Mora [GM89].

L'existence de paramétrisations des idéaux radicaux I de dimension 0 est souvent montrée via la décomposition de R/I en sous-algèbres locales. L'approche utilisée dans ce chapitre avec les éléments primitifs « génériques » remonte aux travaux de Kronecker [Kro82].

L'intérêt pratique des paramétrisations compactes sous forme de Kronecker a été souligné en calcul formel par Rouillier [Rou99]. Son algorithme, appelé RUR (pour *Rational Univariate Representation* en anglais), procède par modules multiples pour reconstruire des paramétrisations à partir de bases standard pour l'ordre tdeg. Une amélioration a été proposée par Bostan, Salvy et Schost [BSS03], en utilisant les méthodes de transposition d'algorithmes linéaires du Chapitre 12.

Bibliographie

Arr06 Arrondo, Enrique (2006). « Another elementary proof of the Nullstellensatz ». In : *The American Mathematical Monthly*, vol. 113, n°2, p. 169–171.

BSS03 Bostan, Alin, Bruno Salvy et Éric Schost (2003). « Fast algorithms for zero-dimensional polynomial systems using duality ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 14, n°4, p. 239–272.

GM89 Gianni, Patrizia et Teo Mora (1989). « Algebraic solution of systems of polynomial equations using Groebner bases ». In : *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes. 5th International Conference, AAECC-5, Menorca, Spain, June 15-19, 1987. Proceedings*. Éd. par Llorenc Huguet et Alain Poli. Vol. 356. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, p. 247–257.

Kro82 Kronecker, Leopold (1882). « Grundzüge einer arithmetischen Theorie der algebraischen Grössen ». In : *Journal für die Reine und Angewandte Mathematik*, vol. 92, p. 1–122.

Rou99 Rouillier, Fabrice (1999). « Solving zero-dimensional systems through the rational univariate representation ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 9, n°5, p. 433–461.

26. Fonction et polynôme de Hilbert, dimension, degré

Résumé

Le polynôme de Hilbert est un objet combinatoire associé à un idéal. On peut lire sur ce polynôme des informations géométriques sur la variété correspondant à l'idéal comme sa dimension et son degré. L'étude de ce polynôme permet aussi d'obtenir une borne de complexité pour les bases standard pour une grande famille de systèmes polynomiaux.

Sauf indication contraire, ce chapitre est placé dans le *contexte projectif*. L'anneau R est donc $k[X_0,\dots,X_n]$.

26.1 Fonction et polynôme de Hilbert

Le *degré* d'un point $a=(a_0,\dots,a_n)$ de $\mathbb{N}^{n+1}$ est l'entier $|a|=a_0+\cdots+a_n$.

Définition 26.1 — Fonction de Hilbert. Soit E une partie stable de $\mathbb{N}^{n+1}$; la fonction HF_E, qui associe à tout entier u le nombre de points de degré u n'appartenant pas à E, est appelée *fonction de Hilbert* de E :

$$\mathrm{HF}_E(u)=\#\{a\in\mathbb{N}^{n+1}\mid a\notin E,|a|=u\}.$$

On étend HF_E aux entiers négatifs par la valeur 0.

Un résultat remarquable, dû à Hilbert, exprime la grande régularité de cette fonction.

Théorème 26.1 — Hilbert. Pour u assez grand, la fonction HF_{E} est égale à un polynôme. Celui-ci est le *polynôme de Hilbert* HP_{E}, et il s'écrit

$$\mathrm{HP}_{\mathrm{E}}(u) = \sum_{i=0}^{d} c_i \binom{u}{d-i}$$

pour des entiers $c_0, \ldots, c_d$ avec $c_0 \neq 0$, où $\binom{u}{r} := \dfrac{u(u-1)(u-2)\cdots(u-r+1)}{r!}$ est la fonction binomiale.

Si u est strictement négatif, alors on a

$$\binom{u}{r} := (-1)^r \frac{(-u)(-u+1)(-u+2)\cdots(-u+r-1)}{r!} = (-1)^r \binom{-u+r-1}{r}.$$

Par conséquent $\binom{u}{r} \in \mathbb{Z}$, et donc $\mathrm{HP}_{\mathrm{E}}(u) \in \mathbb{Z}$ dès que $u \in \mathbb{Z}$. Le coefficient de tête du polynôme HP_{E} est $c_0/d!$. Avant de prouver ce théorème, nous introduisons deux notions qui permettront de relier algèbre et géométrie.

Définition 26.2 Avec les notations précédentes, d est la *dimension* de E, et c_0 son *degré*. Par convention, le degré du polynôme nul sera -1, c'est-à-dire qu'une partie stable de $\mathbb{N}^{n+1}$ qui touche tous les axes de coordonnées a dimension -1.

Définition 26.3 — Régularité et degré maximum. La *régularité* $\mathrm{H}(\mathrm{E})$ de la fonction de Hilbert HF_{E} est le plus petit entier à partir duquel la fonction de Hilbert coïncide avec le polynôme de Hilbert HP_{E}. Si E est non vide, nous notons $\mathrm{D}(\mathrm{E})$ le *degré maximum* des éléments engendrant minimalement E.

Toutes ces définitions s'étendent directement aux idéaux homogènes :

Définition 26.4 Soient $\mathrm{R} = k[\mathrm{X}_0, \ldots, \mathrm{X}_n]$, et $<$ un ordre monomial admissible sur R. On définit alors la *fonction de Hilbert*, la *dimension*, le *degré* et la *régularité* de l'idéal homogène I comme étant ceux de $\mathrm{E}_<(\mathrm{I})$.

La fonction de Hilbert d'un idéal homogène I est aisément calculable à partir d'une base de Gröbner de I. Par ailleurs, nous avons vu au Théorème 23.14, que pour tout entier u, le k-espace vectoriel de R/I, noté $(\mathrm{R}/\mathrm{I})_u$, engendré par les polynômes homogènes de R de degré u est isomorphe à $\bigoplus_{a \notin \mathrm{E}(\mathrm{I}), |a|=u} k\mathrm{X}^a$. Il s'ensuit que $\mathrm{HF}_{\mathrm{R}/\mathrm{I}}(u)$ n'est autre que $\dim(\mathrm{R}/\mathrm{I})_u$.

Une autre observation simple est que si $\mathrm{I} \subset \mathrm{J}$ sont deux idéaux de R, alors $\mathrm{E}_<(\mathrm{I}) \subset \mathrm{E}_<(\mathrm{J})$ et donc $\mathrm{HF}_{\mathrm{E}_<(\mathrm{I})}(u) \geq \mathrm{HF}_{\mathrm{E}_<(\mathrm{J})}(u)$ pour tout u, ce qui entraîne la même inégalité pour leur polynômes de Hilbert et finalement $\dim \mathrm{I} \geq \dim \mathrm{J}$.

R L'hypothèse d'homogénéité est importante pour cette définition : l'escalier $\mathrm{E}_{<(\mathrm{I})}$ dépend de l'ordre admissible, mais sa fonction de Hilbert n'en dépend pas, par le

Théorème 23.14 et la Proposition 23.3. Les mêmes notions peuvent être définies dans le cas affine, et alors il faut de plus supposer que l'ordre monomial est *gradué*, c'est-à-dire que deux monômes de degré total différents sont ordonnés par leur degré. Une autre façon de procéder est de définir la fonction de Hilbert affine d'un idéal comme celle de son homogénéisé.

R Les idéaux de dimension zéro définis au chapitre précédent sans référence à la notion de dimension sont ceux dont le complémentaire de l'escalier est un ensemble de monômes $\{m_1,\dots,m_d\}$ fini. En homogénéisant l'idéal et quitte à choisir un ordre admissible adapté, pour u plus grand que le degré des monômes m_i, les monômes de degré u dans le complémentaire de l'idéal homogénéisé sont les produits $\{m_1 X_0^{u-\deg m_1},\dots,m_d X_0^{u-\deg m_d}\}$, de sorte que la fonction de Hilbert de l'idéal homogénéisé est constante à partir d'un tel u, et donc que l'idéal homogénéisé est de dimension zéro au sens de la définition ci-dessus.

26.2 Démonstrations et borne supérieure de la régularité

Dans cette section, pour simplifier les notations, on pose $e := \mathrm{D(E)}$, et nous prouvons le Théorème 26.1, ainsi que la borne supérieure suivante par récurrence sur n :

$$\mathrm{H(E)} \le (n+1)(e-1)+1. \tag{26.1}$$

Exemple 26.1 Étant donnés $n+1$ entiers positifs $a_0,\dots,a_n$, l'idéal $(X_0^{a_0},\dots,X_n^{a_n})$ a ces générateurs monomiaux comme base standard. La partie stable associée E est le complément d'un parallélépipède dont l'élément de degré maximal est $(a_0-1,\dots,a_n-1)$. La régularité est donc $1+\sum_{i=0}^{n}(a_i-1)$; dans le cas où tous les a_i sont égaux, ceci prouve que la borne (26.1) peut être atteinte.

Démonstration du Théorème 26.1. Pour $n=0$, une partie stable E de $\mathbb{N}$ est de la forme $m+\mathbb{N}$, et $\mathrm{HF_E}(u)$ vaut alors 1 si $u<m$ et 0 sinon, c'est-à-dire que $\mathrm{HP_E}(u)=0$. Le degré D(E) vaut alors m, qui coïncide avec la régularité, et donc l'inégalité (26.1) est bien satisfaite.

Supposons l'hypothèse de récurrence satisfaite jusqu'à $n-1$. Elle s'applique donc aux sections E_i de E par les hyperplans $\{x_n=i\}$. Ces sections sont constantes dès que $i \ge e$. La fonction de Hilbert s'écrit comme somme de ces sections :

$$\mathrm{HF_E}(u) = \sum_{i=0}^{u} \mathrm{HF_{E_i}}(u-i).$$

Cette somme se décompose en trois termes de la façon suivante :

$$\begin{aligned}\mathrm{HF_E}(u) &= \sum_{0\le i\le e-1} \mathrm{HF_{E_i}}(u-i) + \sum_{e\le i\le u-\mathrm{H}(\mathrm{E}_e)} \mathrm{HF_{E_e}}(u-i) + \sum_{u-\mathrm{H}(\mathrm{E}_e)+1\le i\le u} \mathrm{HF_{E_e}}(u-i)\\ &= \sum_{0\le i\le e-1} \mathrm{HF_{E_i}}(u-i) + \sum_{\mathrm{H}(\mathrm{E}_e)\le i\le u-e} \mathrm{HF_{E_e}}(i) + \sum_{0\le i\le \mathrm{H}(\mathrm{E}_e)-1} \mathrm{HF_{E_e}}(i)\end{aligned}$$

$$= \sum_{0\le i\le e-1} \mathrm{HF}_{\mathrm{E}_i}(u-i) + \sum_{\mathrm{H}(\mathrm{E}_e)\le i\le u-e} \mathrm{HP}_{\mathrm{E}_e}(i) + \sum_{0\le i\le \mathrm{H}(\mathrm{E}_e)-1} \mathrm{HF}_{\mathrm{E}_e}(i),$$

où dans la dernière égalité $\mathrm{HF}_{\mathrm{E}_e}(i)$ est devenu $\mathrm{HP}_{\mathrm{E}_e}(i)$ dans la deuxième somme puisque $\mathrm{H}(\mathrm{E}_e) \le i$. La troisième somme ne dépend pas de u et se comporte donc comme une constante. Il suffit alors d'analyser le comportement des deux premières.

L'hypothèse de récurrence implique $\mathrm{H}(\mathrm{E}_i) \le n(e-1)+1$. Par conséquent, dès que $u \ge (n+1)(e-1)+1 \ge \max\{i+\mathrm{H}(\mathrm{E}_i) \mid 0 \le i \le e-1\}$, on a $\mathrm{HF}_{\mathrm{E}_i}(u-i) = \mathrm{HP}_{\mathrm{E}_i}(u-i)$, et donc la première somme coïncide avec le polynôme $\sum_{0\le i\le e-1} \mathrm{HP}_{\mathrm{E}_i}(u-i)$, qui envoie $\mathbb{Z}$ dans $\mathbb{Z}$.

Par ailleurs, le Lemme 26.2 ci-dessous nous assure que la deuxième somme est un polynôme qui envoie $\mathbb{Z}$ dans $\mathbb{Z}$. On en déduit que $\mathrm{HF}_{\mathrm{E}}(u)$ coïncide avec une fonction polynomiale dès que $u \ge (n+1)(e-1)+1$, ce qui conclut la preuve de l'inégalité (26.1).

Lemme 26.2 Soit f un polynôme de $k[\mathrm{X}]$ de degré $d \ge 0$. La fonction F de $\mathbb{Z}$ dans k définie par $\mathrm{F}(i) := \sum_{j=0}^{i} f(j)$ si $i \ge 0$, et $\mathrm{F}(i) := -\sum_{j=i+1}^{-1} f(j)$ si $i < 0$, coïncide avec un polynôme de degré $d+1$.

Démonstration. La preuve peut se faire par récurrence sur d. Si $d = 0$ alors f est une constante et le lemme est vrai. Supposons le lemme vrai jusqu'un degré $d-1 \ge 0$. On peut supposer que f est le monôme X^d. Comme

$$\mathrm{F}(i) - \frac{(i+1)^{d+1}}{d+1} = \sum_{j=0}^{i} \left(j^d - \frac{(j+1)^{d+1}}{d+1} + \frac{j^{d+1}}{d+1} \right), \text{ si } i \ge 0, \text{ et}$$

$$\mathrm{F}(i) - \frac{(i+1)^{d+1}}{d+1} = -\sum_{j=i+1}^{-1} \left(j^d - \frac{(j+1)^{d+1}}{d+1} + \frac{j^{d+1}}{d+1} \right), \text{ si } i < 0,$$

et comme $j^d - \frac{(j+1)^{d+1}}{d+1} + \frac{j^{d+1}}{d+1}$ est un polynôme en j de degré au plus $d-1$, l'hypothèse de récurrence implique que les membres droits des précédentes égalités coïncident avec un même polynôme de degré au plus d. Par conséquent F coïncide bien avec un polynôme de degré $d+1$. ■

Pour conclure la preuve du Théorème 26.1, il ne reste plus qu'à montrer que les coefficients c_i sont des entiers. À nouveau, il suffit de procéder par récurrence, sur u. Comme $\mathrm{HP}_{\mathrm{E}}(0) = c_d$, alors c_d est entier. Ensuite, si i est un entier inférieur ou égal à d on a $\mathrm{HP}_{\mathrm{E}}(i) = \sum_{j=d-i+1}^{d} c_j \binom{i}{d-j} + c_{d-i}$, et donc, comme différence d'entiers, c_{d-i} est un entier aussi. ■

26.3 Dimension et géométrie

Si $\mathrm{V} = \mathbf{V}(\mathrm{I})$ est une variété projective alors sa dimension et son degré sont définis comme étant ceux de I. Dans cette section, nous supposons que k est algébriquement clos, et nous montrons que cette notion de dimension coïncide alors avec plusieurs

interprétations de la dimension en termes géométriques, et que cette dimension se lit aussi sur des escaliers associés à l'idéal.

Définition 26.5 — Dimension de section et de projection. La *dimension de section* est le plus petit entier σ tel qu'il existe un sous-espace linéaire de codimension $\sigma+1$ n'intersectant pas $\mathbf{V}(\mathrm{I})$. La *dimension de projection* est le plus grand entier π tel qu'il existe une projection envoyant surjectivement $\mathbf{V}(\mathrm{I})$ sur un espace de dimension π.

Définition 26.6 — Dimension lexicographique inférieure (resp. supérieure). Soit x un système de coordonnées de $\mathbb{P}^n$. Nous appellerons $linf_x$ (resp. $lsup_x$) le plus petit (resp. le plus grand) entier e tel que $\mathrm{E}_x(\mathrm{I})$ relativement à l'ordre `tdeg` (resp. `lex`) rencontre les derniers $n-e$ axes de coordonnées de $\mathbb{N}^{n+1}$ (resp. ne rencontre pas le plan des $e+1$ premières coordonnées).

Le minimum *linf* des $linf_x$ (resp. le maximum *lsup* des $lsup_x$) pris sur tous les systèmes de coordonnées est appelé la dimension lexicographique inférieure (resp. supérieure) de I.

Théorème 26.3 — Dimension. Pour tout idéal homogène I de $\mathrm{R}=k[\mathrm{X}_0,\dots,\mathrm{X}_n]$ avec k algébriquement clos, la dimension de I coïncide avec sa dimension de section, sa dimension de projection, sa dimension lexicographique inférieure, et sa dimension lexicographique supérieure.

Remarquons que le théorème peut-être vérifié aisément lorsque I est engendré par des polynômes de degré 1, par des techniques classiques d'algèbre linéaire, telles que rappelées au début du prochain chapitre. Par ailleurs, le Théorème 25.7 implique le théorème lorsque I est zéro-dimensionnel.

Démonstration. Notons r la dimension de I, nous allons prouver la suite d'inégalités

$$r \geq lsup \geq \pi \geq \sigma \geq linf \geq r.$$

Tout d'abord pour prouver que $r \geq lsup$, nous supposons qu'il existe des coordonnées de $\mathbb{P}^n$ pour lesquelles, relativement à un ordre monomial admissible, par exemple l'ordre lexicographique, E(I) ne rencontre pas le plan des $e+1$ premières coordonnées. Alors la fonction de Hilbert $\mathrm{HF}_{\mathrm{R}/\mathrm{I}}(u)$ est minorée par le nombre de monômes de degré u en $e+1$ variables, qui est équivalent à $u^e/e!$ quand u tend vers l'infini.

Prouvons maintenant que $lsup \geq \pi$. Supposons que $\mathbf{V}(\mathrm{I})$ puisse être projetée surjectivement sur une base B de dimension b. Nous pouvons choisir des coordonnées telles que B soit définie par les équations $\mathrm{X}_{b+1}=\cdots=\mathrm{X}_n=0$. Ainsi l'idéal $\mathrm{I}\cap k[\mathrm{X}_0,\dots,\mathrm{X}_b]$ se réduit à 0 ; car sinon il existerait un polynôme non nul $f(\mathrm{X}_0,\dots,\mathrm{X}_b)$ dans I qui ne s'annulerait pas sur tout B, et $\mathbf{V}(\mathrm{I})$ ne pourrait pas se projeter surjectivement sur B. Par rapport à de telles coordonnées, la partie stable E(I) relativement à l'ordre `lex` ne peut rencontrer le plan des $b+1$ premières variables d'après le Théorème 24.6.

Montrons maintenant que $\pi \geq \sigma$. Par définition de σ, il existe un sous-espace linéaire L de codimension $\sigma+1$ évitant $\mathbf{V}(\mathrm{I})$. Par ailleurs nous pouvons choisir un

sous-espace linéaire B de dimension σ évitant L. La projection de centre L envoie surjectivement $\mathbf{V}(I)$ sur B, puisque σ est minimal ; car si b est un point de B, le sous-espace linéaire qu'il engendre avec L est de codimension σ, donc coupe $\mathbf{V}(I)$ en au moins un point qui se projette sur b.

Montrons maintenant que $\sigma \geq \mathit{linf}$. Par définition de σ, il existe un sous-espace linéaire L de codimension $\sigma+1$ évitant $\mathbf{V}(I)$. Nous pouvons choisir des coordonnées telles que L soit définie par les équations $X_0 = \cdots = X_\sigma = 0$. L'idéal $J = I + (X_0, \ldots, X_\sigma)$ définit la variété projective vide, donc il contient une puissance de l'idéal maximal $(X_0, \ldots, X_n)$ par le Théorème 25.20. Quelque soit l'ordre, en particulier l'ordre `tdeg`, l'escalier E(J) coupe tous les axes de coordonnées. Considérons un polynôme dont le monôme dominant est sur un des axes $X_{\sigma+1}, \ldots, X_n$; il est congru modulo $(X_0, \ldots, X_\sigma)$ à un polynôme de I, non nul et de même monôme dominant. Par conséquent la section de E(I) par le plan des $n-\sigma$ dernières coordonnées recoupe tous les axes.

Montrons maintenant que $\mathit{linf} \geq r$. Supposons qu'il y ait des coordonnées de $\mathbb{P}^n$ telles que, relativement à l'ordre `tdeg`, l'escalier E(I) rencontre les derniers $n - \mathit{linf}$ axes. En degré u suffisamment grand, le complémentaire de E(I) ne contient que des monômes dont les $n - \mathit{linf}$ derniers exposants sont majorés par un entier fixé. À une constante multiplicative près, la fonction de Hilbert $\mathrm{HF}(u)$ est donc majorée par le nombre de monômes en $\mathit{linf}+1$ variables, soit par $O(u^{\mathit{linf}})$ quand u tend vers l'infini. ■

Corollaire 26.4 Pour tout idéal homogène de R, sa dimension et celle de son radical coïncident.

Démonstration. C'est une conséquence du Nullstellensatz : $\mathbf{V}(\sqrt{I}) = \mathbf{V}(I)$ et donc leurs dimensions de section ou de projection sont identiques. ■

26.4 Position de Noether et degré

Le degré défini à partir du polynôme de Hilbert possède également des interprétations algébriques et géométriques. Le pendant géométrique sera présenté au chapitre suivant ; le point de vue algébrique développé ici repose sur une analyse de bases monomiales des k-espaces vectoriels $(R/I)_u$ (Théorème 23.14), à l'aide de bases standard.

Exemple 26.2 Prenons $n = 2$, et $I = (X_1X_2 - X_0^2)$, et munissons le monoïde des monômes de l'ordre lexicographique induit par $X_0 < X_1 < X_2$. La base standard de I contient l'unique polynôme $X_1X_2 - X_0^2$. En utilisant le Théorème 23.14, on vérifie que $\{X_0, X_1, X_2\}$ est une base de $(R/I)_1$, puis que $\{X_0^2, X_0X_1, X_1^2, X_0X_2, X_2^2\}$ est une base de $(R/I)_2$ (il s'agit de tous les monômes de degré 2 sauf X_1X_2). Lorsque $u \geq 3$, l'espace $(R/I)_u$ est engendré par tous les monômes de degré u qui ne sont pas des multiples de X_1X_2, c'est-à-dire X_0^u, les $X_0^{u-i}X_1^i$ pour $1 \leq i \leq u$, et les $X_0^{u-i}X_2^i$ pour $1 \leq i \leq u$. La fonction de Hilbert vaut alors $2u+1$. La dimension de I est 1 et son degré 2.

Dans cette section nous allons montrer que des bases des $(R/I)_u$ peuvent être décrites aisément, lorsque u est assez grand et que les variables $X_0,\dots,X_n$ satisfont certaines propriétés pour I, qui motivent les prochaines définitions.

Définition 26.7 Soient $A \subseteq B$ deux anneaux. On dit que $b \in B$ est *entier* sur A s'il existe $P \in A[X]$ unitaire tel que $P(b) = 0$. Un tel polynôme P est appelé une *relation de dépendance entière* pour b. L'extension $A \subseteq B$ est *entière* si tout $b \in B$ est entier sur A.

Exemple 26.3 L'anneau $\mathbb{Z}[\sqrt{2}] = \mathbb{Z}[X]/(X^2-2)$ est une extension entière de $\mathbb{Z}$: tout élément de $\mathbb{Z}[\sqrt{2}]$ s'écrit $a\sqrt{2}+b$ pour deux entiers a et b, et un polynôme annulant ce nombre est donné par le résultant de $X-(aT+b)$ et T^2-2, qui vaut $X^2-2bX+b^2-2a^2$ et est unitaire.

Définition 26.8 Un idéal I homogène de R est dit en *position de Noether* s'il existe $r \in \{0,\dots,n\}$ tel que $I \cap k[X_0,\dots,X_r] = (0)$, et R/I est une extension entière de $k[X_0,\dots,X_r]$.

Ainsi, en position de Noether, chacune des variables $X_{r+1},\dots,X_n$ est entière sur $k[X_0,\dots,X_r]$ modulo I, ou plus explicitement, pour chaque $i \in \{r+1,\dots,n\}$, il existe dans I un polynôme unitaire en X_i et ne faisant pas intervenir les autres variables de $\{X_{r+1},\dots,X_n\}$.

Exemple 26.4 L'idéal $(X_2^2 - X_0X_1)$ de $k[X_0,X_1,X_2]$ est en position de Noether avec $r=1$. En revanche, l'idéal $(X_1X_2 - X_0^2)$ ne l'est pas : s'il l'était il contiendrait un polynôme unitaire en X_2 à coefficients dans $k[X_0,X_1]$, mais l'évaluation en $X_0=1, X_1=0$ de ce polynôme mène à une contradiction.

Exemple 26.5 Prenons $n=2$, et $I=(X_2^2-X_0X_1)$, et munissons à nouveau le monoïde des monômes de l'ordre lexicographique induit par $X_0 < X_1 < X_2$. La base standard de I contient l'unique polynôme $X_2^2 - X_0X_1$. En utilisant à nouveau le Théorème 23.14, on vérifie que $\{X_0,X_1,X_2\}$ est une base de $(R/I)_1$, puis que $\{X_0^2, X_0X_1, X_1^2, X_0X_2, X_1X_2\}$ est une base de $(R/I)_2$ (il s'agit de tous les monômes de degré 2 sauf X_2^2). Lorsque $u \geq 3$, l'espace $(R/I)_u$ est engendré par tous les monômes de degré u qui ont un degré partiel en X_2 qui vaut 0 ou 1, c'est-à-dire les $X_0^{u-i}X_1^i$ pour $0 \leq i \leq u$, et les $X_0^{u-1-i}X_1^iX_2$ pour $0 \leq i \leq u-1$. La fonction de Hilbert vaut alors $2u+1$. La dimension de I est 1 et son degré 2.

Théorème 26.5 Soit I un idéal homogène de $R = k[X_0,\dots,X_n]$ en position de Noether. On note $K_r = k(X_0,\dots,X_r)$ et I' l'extension de I dans $K_r[X_{r+1},\dots,X_n]$. Alors, I' est un idéal non homogène de $K_r[X_{r+1},\dots,X_n]$ qui est de dimension zéro au sens de la Définition 25.2. De plus, la dimension de I est r, et la dimension de la K_r-algèbre $K_r[X_{r+1},\dots,X_n]/I'$ est égale à $\deg I$.

Démonstration. On pose $A_r = k[X_0, \dots, X_r]$. Pour la première assertion, il suffit de vérifier la condition (i) du Théorème 25.7 sur les idéaux de dimension 0. Tout d'abord I′ est propre. Autrement il contiendrait 1, ce qui impliquerait l'existence d'un polynôme non nul dans $I \cap A_r$, ce qui est impossible du fait de la position de Noether. D'autre part, l'existence de relations de dépendance entière pour chaque variable $X_{r+1}, \dots, X_n$ garantit que le complémentaire de l'escalier de I′ est fini pour tout ordre admissible sur les variables $X_{r+1}, \dots, X_n$.

On munit le monoïde des monômes de l'ordre lexicographique induit par $X_0 < X_1 < \cdots < X_n$, et on note juste E l'escalier de I pour cet ordre. On introduit la projection π de $\mathbb{N}^{n+1}$ vers $\mathbb{N}^{n-r}$ qui ne conserve que les $(n-r)$-ièmes dernières coordonnées, c'est-à-dire :

$$\begin{array}{rccc} \pi : & \mathbb{N}^{n+1} & \to & \mathbb{N}^{n-r} \\ & (a_0, \dots, a_n) & \mapsto & (a_{r+1}, \dots, a_n). \end{array}$$

L'image $\pi(E)$ est une partie stable de $\mathbb{N}^{n-r}$. Le fait que les variables $X_{r+1}, \dots, X_n$ soient entières sur $k[X_0, \dots, X_r]$ modulo I et que l'on utilise l'ordre lexicographique, implique que le complémentaire de $\pi(E)$ dans $\mathbb{N}^{n-r}$ est un ensemble fini noté $\{e_1, \dots, e_\delta\}$. Nous allons montrer que ce nombre δ est d'une part le degré de I, et d'autre part la dimension de $K_r[X_{r+1}, \dots, X_n]/I'$. Le monôme correspondant à $e_i = (e_{i,r+1}, \dots, e_{i,n})$ sera noté $m_i = X_{r+1}^{e_{i,r+1}} \cdots X_n^{e_{i,n}}$. Remarquons que pour tout $i \in \{1, \dots, \delta\}$ et pour tout $(a_0, \dots, a_r) \in \mathbb{N}^r$, l'exposant $(a_0, \dots, a_r, e_{i,r+1}, \dots, e_{i,n})$ n'est pas dans E.

Notons $\tilde{E} = \mathbb{N}^{n+1} \setminus E$ l'ensemble des monômes sous l'escalier, et examinons sa projection $\pi(\tilde{E})$. La position de Noether garantit à nouveau qu'il s'agit d'un ensemble fini. Comme E est une partie stable, un exposant $(a_{r+1}, \dots, a_n)$ appartient à cette projection si, et seulement si, $(0, \dots, 0, a_{r+1}, \dots, a_n)$ est dans $\tilde{E}$. Cela donne l'inclusion

$$\mathbb{N}^{r+1} \setminus \pi(E) \subseteq \pi(\tilde{E}). \tag{26.2}$$

L'Exemple 26.6 ci-après montre que cette inclusion peut être stricte. Les exposants de $\pi(\tilde{E})$ hors de $\mathbb{N}^{r+1} \setminus \pi(E)$ seront notés $\tilde{e}_i = (\tilde{e}_{i,r+1}, \dots, \tilde{e}_{i,n})$ pour i de 1 à s. Le monôme correspondant sera noté $\tilde{m}_i = X_{r+1}^{\tilde{e}_{i,r+1}} \cdots X_n^{\tilde{e}_{i,n}}$. Par construction, pour chaque $\tilde{e}_i$, il existe $\dot{e}_i = (\dot{e}_{i,0}, \dots, \dot{e}_{i,r})$ tel que $(\dot{e}_{i,0}, \dots, \dot{e}_{i,r}, \tilde{e}_{i,r+1}, \dots, \tilde{e}_{i,n})$ appartienne à E. Le monôme correspondant sera noté $\dot{m}_i = X_0^{\dot{e}_{i,0}} \cdots X_r^{\dot{e}_{i,r}}$.

La valeur en $u \in \mathbb{N}$ de la fonction de Hilbert de E est le cardinal de $C_u = \{a \in \tilde{E} \mid |a| = u\}$, la partie de degré u du complémentaire de E. En posant B_u l'ensemble des vecteurs de la forme $(a_0, \dots, a_r, e_{i,r+1}, \dots, e_{i,n})$ avec $i \in \{1, \dots, \delta\}$ et $a_0 + \cdots + a_r + |e_i| = u$, nous allons montrer que le cardinal de C_u est équivalent à celui de B_u lorsque u tend vers l'infini. La remarque suivant la définition de m_i montre que B_u est inclus dans C_u, et on a

$$\operatorname{card}(B_u) = \sum_{i=1}^{\delta} \binom{u - \deg m_i + r}{r} = \delta \frac{u^r}{r!} + O(u^{r-1}),$$

lorsque u est suffisamment grand. La projection d'un exposant e de $C_u \setminus B_u$ est par construction l'un des $\tilde{e}_i$ pour $i \in \{1, \dots, s\}$. Pour $i \in \{1, \dots, s\}$, notons

$$\tilde{M}_{i,u} = \{(a_0, \dots, a_r, \tilde{e}_{i,r+1}, \dots, \tilde{e}_{i,n}) \mid (a_0, \dots, a_r) \in \mathbb{N}^{r+1}, a_0 + \cdots + a_r + |\tilde{e}_i| = u\}.$$

Nous avons

$$C_u \setminus B_u = \cup_{i=1}^{s} (\tilde{M}_{i,u} \cap C_u).$$

Par ailleurs, pour tout $i \in \{1,\dots,s\}$, on a

$$\tilde{M}_{i,u} \cap C_u \subseteq \tilde{M}_{i,u} \setminus \tilde{N}_{i,u},$$

où

$$\tilde{N}_{i,u} = \{(\dot{e}_{i,0} + a_0, \dots, \dot{e}_{i,r} + a_r, \tilde{e}_{i,r+1}, \dots, \tilde{e}_{i,n}) \mid (a_0,\dots,a_r) \in \mathbb{N}^{r+1}, a_0 + \cdots + a_r + |\dot{e}_i| + |\tilde{e}_i| = u\}.$$

En utilisant $\operatorname{card}(\tilde{M}_{i,u}) = u^r/r! + O(u^{r-1})$ et $\operatorname{card}(\tilde{N}_{i,u}) = u^r/r! + O(u^{r-1})$, on obtient $\operatorname{card}(\tilde{M}_{i,u} \cap C_u) = O(u^{r-1})$, et enfin $\operatorname{card}(C_u) = \operatorname{card}(B_u) + O(u^{r-1})$. Nous venons donc de prouver

$$\mathrm{HF}_E(u) = \delta \frac{u^r}{r!} + O(u^{r-1}).$$

Par le Théorème 26.1, il s'ensuit que r est la dimension de I, et que δ en est son degré.

Pour conclure la preuve, nous allons montrer que $m_1,\dots,m_\delta$ est une base de $K_r[X_{r+1},\dots,X_n]/I'$. On suppose maintenant les $\tilde{m}_i$ ordonnés de façon croissante $\tilde{m}_1 < \cdots < \tilde{m}_s$. Tout élément de $K_r[X_{r+1},\dots,X_n]/I'$ s'écrit sous la forme P/p avec $P \in R$ et $p \in A_r$. Par le Théorème 23.14, P se récrit modulo I sous la forme d'un polynôme de $\sum_{i=1}^{\delta} A_r m_i + \sum_{i=1}^{s} A_r \tilde{m}_i$. En particulier, chaque monôme $\dot{m}_i \tilde{m}_i$ est le monôme de tête d'un polynôme P_i de I qui appartient à $\sum_{i=1}^{\delta} A_r m_i + \sum_{j=1}^{i} A_r \tilde{m}_j$. Autrement dit, modulo I' chaque monôme $\tilde{m}_i$ se récrit en un polynôme de $\sum_{i=1}^{\delta} K_r m_i + \sum_{j=1}^{i-1} K_r \tilde{m}_j$. Cela montre que les m_i forment une famille génératrice de $K_r[X_{r+1},\dots,X_n]/I'$.

Soit maintenant une relation $\sum_{i=1}^{\delta} c_i m_i \in I'$, avec les c_i dans K_r. Il existe un polynôme non nul $p \in A_r$ tel que tous les pc_i soient dans A_r et $P = \sum_{i=1}^{\delta} pc_i m_i$ soit dans I. Tous les monômes de P sont dans le complémentaire de E et donc P est nul. Il s'ensuit que les c_i sont tous nuls. Finalement les m_i forment une famille libre de $K_r[X_{r+1},\dots,X_n]/I'$. ■

Exemple 26.6 Avec $n = 2$, l'idéal $I = (X_2^2, X_1X_2) \subseteq k[X_0,X_1,X_2]$ est en position de Noether. On vérifie $\dim I = 1$ et $\deg I = 1$ en calculant son polynôme de Hilbert. Avec les notations de la preuve on a : $I' = (X_2)$, $\pi(E)$ est la partie engendrée par (1) (l'exposant du monôme X_2), et $\pi(\tilde{E})$ est l'ensemble des deux vecteurs $\{(0),(1)\}$ (les exposants des monômes 1 et X_2).

Le chapitre suivant est consacré à l'étude et au calcul des positions de Noether, des points de vue de l'algèbre et de la géométrie. Nous y verrons en particulier une interprétation géométrique du degré d'une variété projective.

26.5 Suites régulières

Les suites régulières fournissent une classe très importante de systèmes polynomiaux qui sont très communs en pratique, sur lesquels les calculs de bases standard se déroulent bien et pour lesquels on dispose d'une formule explicite pour la série de Hilbert.

Définition 26.9 Soit R un anneau. On dit qu'un élément $a \in R$ est un *diviseur de zéro* dans R s'il existe $b \in R \setminus \{0\}$ tel que $ab = 0$. Ceci est équivalent au fait que l'endomorphisme de multiplication par a dans R ne soit pas injectif.

L'impact sur la fonction de Hilbert de l'ajout d'un polynôme f à un idéal I dépend de son caractère diviseur de zéro ou non : si J est l'idéal $I + (f) \supset I$, de sorte que R/J est un sous-espace vectoriel de R/I, soit ϕ_j l'application linéaire de $(R/I)_{j-d}$ dans $(R/I)_j$ qui associe à un élément de degré $j-d$ de R/I son produit par f. Son image est formée des éléments de $(R/I)_j$ qui sont équivalents à 0 dans $(R/J)_j$. Elle a donc dimension $\dim(R/I)_j - \dim(R/J)_j$, qui vaut par ailleurs $\dim(R/I)_{j-d} - \dim\ker\phi_j$. Les dimensions de ces espaces satisfont donc $\dim(R/J)_j = \dim(R/I)_j - \dim(R/I)_{j-d} + \dim\ker\phi_j$ ou encore

$$\mathrm{HF}_{R/J}(u) = \mathrm{HF}_{R/I}(u) - \mathrm{HF}_{R/I}(u-d) + \dim\ker\phi_j. \tag{26.3}$$

Pour d assez grand, $\mathrm{HF}_{R/I}(u)$ est un polynôme $\mathrm{HP}_{R/I}(u)$ et $\dim\ker\phi_j$ est quant à lui borné par $\dim(R/I)_{j-d} = \mathrm{HP}_{R/I}(u-d)$. Si les premiers coefficients du polynôme $\mathrm{HP}_{R/I}$ s'écrivent $\mathrm{HP}_{R/I}(u) = \delta\frac{u^r}{r!} + cu^{r-1} + \cdots$, où r est la dimension, δ le degré, et $c \in \mathbb{Q}$, alors

$$\begin{aligned}\mathrm{HP}_{R/I}(u) - \mathrm{HP}_{R/I}(u-d) &= \frac{\delta}{r!}\left(u^r - (u-d)^r\right) + c\left(u^{r-1} - (u-d)^{r-1}\right) + \cdots \\ &= d\delta\frac{u^{r-1}}{(r-1)!} + c'u^{r-2} + \cdots.\end{aligned} \tag{26.4}$$

Une première observation est donc que la dimension chute d'au plus un.

Lemme 26.6 Soit I un idéal homogène de R de dimension r et f un polynôme de R. Alors la dimension de $I + (f)$ est $r-1$ ou r.

On obtient un résultat plus précis dans le cas où f n'est pas diviseur de zéro dans R/I.

Théorème 26.7 Soit I un idéal homogène de dimension r, de degré δ, et de régularité e. Soit f un polynôme homogène de degré d non diviseur de zéro dans R/I. Alors $I + (f)$ a pour dimension $r-1$, pour degré $d\delta$, et sa régularité est bornée par $e + d$.

Démonstration. Comme f n'est pas diviseur de 0, l'application ϕ_j est injective, l'équation (26.3) montre que la régularité de R/J est bornée par $e + d$, et l'équation (26.4) donne le polynôme de Hilbert de R/J, ce qui conclut la preuve. ∎

Définition 26.10 Une suite $f_1, \ldots, f_s$ de polynômes est dite *régulière* si pour tout $i \in \{1, \ldots, s\}$, la classe de f_i dans $R/(f_1, \ldots, f_{i-1})$ n'est pas un diviseur de zéro.

Corollaire 26.8 — Borne de Macaulay. Soit I un idéal homogène engendré par une suite régulière homogène $f_1,\dots,f_s$ de degrés respectifs $d_1,\dots,d_s$. Alors I a pour dimension $n-s$, pour degré $\prod_{i=1}^{s} d_i$ et une régularité majorée par $(\sum_{i=1}^{s} d_i)-n$.

Démonstration. Si $s=0$ alors R/I = R et

$$\mathrm{HF}_{\mathrm{R}}(u)=\begin{cases}\binom{u+n}{n} & \text{pour } u \text{ positif}\\ 0 & \text{pour } u \text{ négatif.}\end{cases}$$

La dimension de R est donc n, son degré est 1, et sa régularité est $-n$ car HF_{R} coïncide avec $\mathrm{HP}_{\mathrm{R}}(u)=\binom{u+n}{n}$ si, et seulement si, $u\geq -n$. Le résultat est finalement une conséquence du théorème précédent. ■

Exercice 26.1 On appelle *série de Hilbert* la série génératrice $\sum_{u\geq 0}\mathrm{HF}_{\mathrm{E}}(u)t^u$. Avec les mêmes hypothèses que dans ce corollaire, calculer la série de Hilbert de l'idéal à l'aide de la récurrence (26.3). ■

Un premier résultat de complexité découle de ces considérations, et sert de base à un résultat plus général au chapitre suivant.

Corollaire 26.9 Si I est un idéal engendré par $f_0,\dots,f_n$ une suite régulière de polynômes homogènes de R avec f_i de degré d_i pour $i\in\{0,\dots,n\}$, alors il admet une base standard minimale dont le degré des éléments est majoré par $(\sum_{i=0}^{n} d_i)-n$.

Démonstration. D'après le résultat précédent, la dimension de I est -1, ce qui signifie que son polynôme de Hilbert est nul, et donc que pour d plus grand que la régularité, tous les polynômes sont dans l'idéal. ■

En termes informels, un système polynomial « bien posé » est une suite régulière. Plus précisément, cette propriété s'énonce comme une réciproque partielle du Corollaire 26.8, dont la preuve dépasse légèrement le cadre de cet ouvrage.

Théorème 26.10 Si $f_1,\dots,f_s$ (avec $1\leq s\leq n+1$) est une suite de polynômes homogènes de R tels que la dimension de $(f_1,\dots,f_s)$ est $n-s$, alors c'est une suite régulière.

Nous finissons ce chapitre en montrant que, dans un anneau de polynômes, une suite régulière, non nécessairement homogène, reste régulière lorsqu'on étend le corps des coefficients. Il est aisé de vérifier cette propriété à partir des définitions si on étend le corps de base avec une ou plusieurs nouvelles variables. En général, la preuve est un peu moins aisée.

Proposition 26.11 Soient I un idéal de R, et f un non diviseur de zéro dans R/I. Soit K une extension de k, et notons I' l'extension de I dans $\mathrm{K}[\mathrm{X}_0,\dots,\mathrm{X}_n]$. Alors, f est non diviseur de zéro dans $\mathrm{K}[\mathrm{X}_0,\dots,\mathrm{X}_n]/\mathrm{I}'$.

Démonstration. Pour tester si un polynôme non nul f est diviseur de zéro dans R/I, on considère l'ensemble $J = \{g/f \mid g \in I \cap (f)\}$, qui est en fait un idéal qui contient I. Supposons que J coïncide avec I, et soit $g \in R$ tel que $gf \in I$. Alors, g appartient à J et donc à I. Il s'ensuit que f est non diviseur de zéro dans R/I. Inversement, supposons que f n'est pas un diviseur de zéro dans R/I, et soit g un élément de J. Par construction on a $gf \in I$ et donc $g \in I$. Finalement on vient de montrer que J coïncide avec I si, et seulement si, f est non diviseur de zéro dans R/I.

Une fois fixé un ensemble de polynômes générateurs $f_1,\dots,f_s$ de I, une base standard $g_1,\dots,g_r$ de $I \cap (f)$ s'obtient par l'algorithme de l'Exercice 24.3. Les polynômes $g_1/f,\dots,g_r/f$ engendrent J, et un nouveau calcul de base standard permet de tester si J est contenu dans I. Par conséquent, tester si f divise zéro dans R/I peut se faire avec un algorithme qui n'effectue des opérations que dans le corps engendré par les coefficients de $f_1,\dots,f_s$ et f. Étendre le corps k dans lequel ces coefficients sont donnés ne change donc rien au calcul et à son résultat. ■

Corollaire 26.12 Soit $f_1,\dots,f_s$ une suite régulière de R. Si K est une extension de k, alors $f_1,\dots,f_s$ est régulière dans $K[X_0,\dots,X_n]$.

Démonstration. C'est une conséquence immédiate de la proposition précédente, par récurrence. ■

26.6 Bases standard pour les suites régulières en position de Noether

Dans la situation fréquente d'idéaux engendrés par une suite régulière en position de Noether, la complexité du calcul de bases standard pour l'ordre `tdeg` est maîtrisée. Le chapitre suivant montrera comment se ramener à la position de Noether si besoin.

Lemme 26.13 Si les polynômes homogènes $f_1,\dots,f_s$ de R forment une suite régulière en position de Noether, alors $f_1,\dots,f_s,X_{n-s},\dots,X_0$ est également une suite régulière.

Démonstration. La preuve repose sur le Théorème 26.10, que nous venons d'admettre. Les variables $X_{n-s+1},\dots,X_n$ étant entières modulo $I := (f_1,\dots,f_s)$, pour chaque $i \in \{n-s+1,\dots,n\}$, il existe dans I un polynôme g_i unitaire en X_i, et ne faisant pas intervenir les autres variables de $\{X_{n-s+1},\dots,X_n\}$. L'idéal étant homogène, on peut supposer ces polynômes homogènes aussi. La spécialisation du polynôme g_i en $X_{n-s} = \cdots = X_0 = 0$ est donc un monôme $X_i^{m_i}$ et il s'ensuit que la dimension de $(f_1,\dots,f_s,X_{n-s},\dots,X_0)$ est -1. La conclusion découle du Théorème 26.10. ■

Nous considérons l'ordre noté $\mathtt{tdeg}(X_n,\dots,X_0)$, variante de l'ordre lexicographique renversé gradué de la Définition 23.17, mais sur les variables permutées $X_0 > X_1 > \cdots > X_n$. Autrement dit, à même degré, les plus grands monômes sont ceux qui ont le moins de X_0, puis le moins de X_1 et ainsi de suite. Par exemple, pour cet ordre, on a

$$X_3^3 > X_3^2X_2 > X_3X_2^2 > X_2^3 > X_3^2X_1 > X_3X_2X_1 > X_2^2X_1 > X_3X_1^2 > X_2X_1^2 > X_1^3.$$

Lemme 26.14 Si I est un idéal engendré par des polynômes homogènes $f_1,\dots,f_s$ de R, qui forment une suite régulière en position de Noether, alors les monômes de tête de toute base standard minimale pour l'ordre $\mathtt{tdeg}(X_n,\dots,X_0)$ ne sont divisibles par aucune des variables $X_0,\dots,X_{n-s}$.

Nous pouvons visualiser ce résultat en disant d'une manière imagée que l'escalier de l'idéal est *cylindrique* le long des variables $X_0,\dots,X_{n-s}$. Autrement dit l'escalier est engendré par des monômes vivant dans sa section par les variables $X_0,\dots,X_{n-s}$.

Démonstration. Il suffit de montrer que si $f \in \mathrm{I} := (f_1,\dots,f_s)$ a un monôme de tête M divisible par l'un des X_i, $i \in \{0,\dots,n-s\}$, alors il existe $g \in \mathrm{I}$ dont le monôme de tête divise M et ne dépend pas de $X_0,\dots,X_{n-s}$. Sans perte de généralité on peut supposer f homogène (Proposition 23.3). Soit ℓ minimal tel que X_ℓ divise M et soit $\alpha_\ell \geq 1$ l'exposant maximal tel que $X_\ell^{\alpha_\ell}$ divise M. Pour cet ordre, si m est un monôme de degré inférieur à α_ℓ en X_ℓ tel que $m < \mathrm{M}$, alors il est divisible par un X_i avec $i < \ell$. Il existe donc des polynômes homogènes $g_0,\dots,g_\ell$ tels que

$$f = X_\ell^{\alpha_\ell} g_\ell + X_{\ell-1} g_{\ell-1} + \cdots + X_0 g_0,$$

g_ℓ appartenant à $k[X_\ell,\dots,X_n]$ et ayant pour monôme de tête $\mathrm{M}/X_\ell^{\alpha_\ell} \in k[X_{\ell+1},\dots,X_n]$.

À ce stade, $X_\ell^{\alpha_\ell} g_\ell \in \tilde{\mathrm{I}} := (f_1,\dots,f_s,X_{\ell-1},\dots,X_0)$. Si $\ell > n-s$, M n'est divisible par aucune des variables $X_0,\dots,X_{n-s}$. Sinon, par le lemme précédent, X_ℓ n'est pas diviseur de 0 modulo I et donc g_ℓ appartient à la fois à l'idéal $\tilde{\mathrm{I}}$ et à l'anneau $k[X_\ell,\dots,X_n]$, et donc à I. La conclusion découle d'une récurrence sur ℓ, puisque $\mathrm{tt}(g)$ divise $\mathrm{tt}(f)$, $g \in \mathrm{I}$ et $\mathrm{tt}(g)$ ne fait plus intervenir $X_0,\dots,X_\ell$. ■

Théorème 26.15 Si I est un idéal homogène de R engendré par une suite régulière $f_1,\dots,f_s$ en position de Noether, alors I admet une base standard minimale pour l'ordre $\mathtt{tdeg}(X_n,\dots,X_0)$ dont les éléments ont degré borné par $\mathrm{D} := \sum_{i=1}^{s}(d_i-1)+1$ et qui peut être calculée en

$$\mathrm{O}\left(s\mathrm{D}\binom{n+\mathrm{D}}{\mathrm{D}}^{\theta}\right)$$

opérations arithmétiques dans k, où θ est un exposant réalisable pour la multiplication de matrices.

Démonstration. D'après le lemme précédent, l'idéal I admet une base standard minimale $g_1,\dots,g_u$ pour l'ordre $\mathtt{tdeg}(X_n,\dots,X_0)$ dont les monômes de tête ne sont pas divisibles par $X_0,\dots,X_{n-s}$. Le Théorème 24.8 montre qu'alors $g_1,\dots,g_u,X_{n-s},\dots,X_0$ est une base standard de l'idéal engendré par la suite $f_1,\dots,f_s,X_{n-s},\dots,X_0$, qui est nécessairement minimale, et qui est régulière d'après le Lemme 26.13. Le Corollaire 26.9 montre que l'idéal qu'elle engendre admet une base standard minimale dont les éléments ont degré au plus $d_1+\cdots+d_s-s+1$. Le résultat de complexité découle alors de l'algorithme naïf par algèbre linéaire sur la matrice de Macaulay (présentée en Section 24.1), et sa complexité est donnée par la Proposition 24.1. ■

Pour mieux apprécier le comportement asymptotique de la borne du théorème ci-dessus, il suffit d'utiliser la formule de Stirling

$$\log \mathrm{N}! = \mathrm{N}\log \mathrm{N} - \mathrm{N} + \frac{1}{2}\log \mathrm{N} + \mathrm{O}(1), \qquad \mathrm{N} \to \infty.$$

Si D et n tendent tous deux vers l'infini, le logarithme du coefficient binomial devient

$$(\mathrm{D}+n)\log(\mathrm{D}+n) - \mathrm{D}\log \mathrm{D} - n\log n + \frac{1}{2}\log\frac{\mathrm{D}+n}{n\mathrm{D}} + \mathrm{O}(1).$$

En posant $n = \lambda s$ avec $\lambda \geq 1$ et en notant δ la moyenne arithmétique des degrés d_i, le degré maximum D devient $s(\delta-1)+1$, et le comportement du logarithme ci-dessus se simplifie : les termes dominants en $n\log n$ ou $s\log s$ disparaissent et il reste

$$s((\lambda+\delta-1)\log(\lambda+\delta-1) - \lambda\log\lambda - (\delta-1)\log(\delta-1)) - \frac{1}{2}\log s + \mathrm{O}(1).$$

Ainsi le comportement de la complexité est simplement exponentiel, et nous avons obtenu.

Corollaire 26.16 Dans les conditions du théorème précédent, si $\delta > 1$ borné est la moyenne arithmétique des degrés d_i et $\lambda = n/s \geq 1$, alors la complexité du théorème devient

$$\mathrm{O}\left(\left(\frac{(\lambda+\delta-1)^{\lambda+\delta-1}}{\lambda^{\lambda}(\delta-1)^{\delta-1}}\right)^{\theta s} s^{2-\theta/2}\right), \qquad s \to \infty.$$

Ce résultat ne s'applique pas lorsque $\delta = 1$, qui correspond à un système linéaire, et où la complexité n'est pas exponentielle.

R Le calcul de bases standard en homogène est mieux compris car le degré des polynômes dans l'algorithme est croissant. Dans le cas affine, l'évolution du degré des polynômes dans l'algorithme est erratique. Cependant, les résultats de complexité de cette section s'appliquent sur la version homogénéisée d'un système inhomogène, si elle reste une suite régulière en position de Noether.

Notes

Ce que nous appelons fonction de Hilbert de I est souvent appelé plutôt fonction de Hilbert de R/I, mais nous suivons l'usage plus simple d'emploi de l'ouvrage de Cox, Little, O'Shea [CLO96].

L'interprétation du degré d'un idéal en termes géométriques dépasse un peu le cadre de cet ouvrage. Pour un idéal de dimension zéro, il s'agit du nombre de points qui annulent l'idéal, en comptant leurs multiplicités. Dans le cas d'un idéal homogène I engendré par une suite régulière $f_1,\ldots,f_n$, et en l'absence de multiplicités, nous verrons dans le Chapitre 28 que le produit des degrés $\deg(f_1)\cdots\deg(f_n)$ majore le nombre de points de $\mathbf{V}(\mathrm{I})$. Ce produit des degrés est appelé une *borne de Bézout* du nombre de solutions du système $f_1 = \cdots = f_n = 0$.

Le Théorème 26.10 est dû à Macaulay [Mac94]. Une preuve dans le langage moderne s'obtient en utilisant la notion de décomposition primaire des idéaux [Lej84, Chap. III, Prop. 2.4].

L'étude des bases standard a conduit à une vaste littérature concernant leurs propriétés et les algorithmes pour les calculer. Il a été prouvé que les degrés des polynômes d'une base standard minimale de $(f_1,\dots,f_s) \subseteq k[X_1,\dots,X_n]$ sont au plus $2(d^2/2+d)^{2^{n-1}}$ lorsque $\deg f_i \leq d$ pour tout i. Cette borne est doublement exponentielle dans le nombre de variables. En outre, il existe des idéaux pour lesquels les bases standard contiennent au moins $2^{2^{cn}}$ éléments et des éléments de degré au moins $2^{2^{cn}}$ pour une certaine constante réelle c [Dem87 ; Huỳ86 ; MM82]. Un résumé des pires cas se trouve dans le livre de von zur Gathen et Gerhard [GG03, Chapitre 21].

La borne de complexité dans le pire cas n'est heureusement pas atteinte de façon systématique. De plus, il est difficile de prévoir le temps de calcul d'une base standard. Néanmoins, pour les systèmes en n variables admettant un nombre fini de solutions (dans une clôture algébrique du corps de base), alors Giusti, Lazard et Lakshman ont conçu des algorithmes de coût $d^{O(n)}$, où d est une borne sur le degré total des équations [Giu84 ; Lak90 ; Lak91 ; Laz83 ; LL91].

Le Théorème 26.15 montre que ce type de complexité simplement exponentielle est accessible pour une large classe de systèmes. Sa présentation provient d'idées dues à Lejeune-Jalabert [Lej84]. Ce théorème repose uniquement sur le degré maximal des éléments de la base et l'algèbre linéaire sur la matrice de Macaulay. Bardet, Faugère et Salvy [BFS15] ont donné une description plus précise de la forme de la base en explicitant le nombre d'éléments de chaque degré. Ils ont aussi conçu un algorithme (une variante de l'algorithme F5 [Fau02]) qui tire parti de la structure de cette matrice pour atteindre une complexité inférieure à celle du Théorème 26.15.

Bibliographie

BFS15 Bardet, Magali, Jean-Charles Faugère et Bruno Salvy (2015). « On the complexity of the F_5 Gröbner basis algorithm ». In : *Journal of Symbolic Computation*, vol. 70, p. 49–70.

CLO96 Cox, David, John Little et Donal O'Shea (1996). *Ideals, varieties, and algorithms. An introduction to computational algebraic geometry and commutative algebra*. 2e éd. Undergraduate Texts in Mathematics. Springer-Verlag.

Dem87 Demazure, M. (1987). « Le théorème de complexité de Mayr et Meyer ». In : *Géométrie algébrique et applications, I (La Rábida, 1984)*. Vol. 22. Travaux en Cours. Paris : Hermann, p. 35–58.

Fau02 Faugère, Jean-Charles (2002). « A new efficient algorithm for computing Gröbner bases without reduction to zero (F5) ». In : *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*. ISSAC'02. New York, NY, USA : ACM, p. 75–83.

GG03 Gathen, Joachim von zur et Jürgen Gerhard (2003). *Modern computer algebra*. 2e éd. Cambridge University Press.

Giu84 Giusti, M. (1984). « Some effectivity problems in polynomial ideal theory ». In : *EUROSAM 84 : International Symposium on Symbolic and Algebraic*

Computation Cambridge, England, July 9–11, 1984. Éd. par John Fitch. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 159–171.

Huỳ86 Huỳnh, Dũng T. (1986). « A superexponential lower bound for Gröbner bases and Church-Rosser commutative Thue systems ». In : *Information and Control*, vol. 68, n°1-3, p. 196–206.

Lak90 Lakshman, Y. N. (1990). « On the complexity of computing a Gröbner basis for the radical of a zero dimensional ideal ». In : *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*. STOC '90. New York, NY, USA : ACM, p. 555–563.

Lak91 — (1991). « A single exponential bound on the complexity of computing Gröbner bases of zero dimensional ideals ». In : *Effective Methods in Algebraic Geometry*. Éd. par Teo Mora et Carlo Traverso. Boston, MA : Birkhäuser Boston, p. 227–234.

Laz83 Lazard, D. (1983). « Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations ». In : *Computer Algebra : EUROCAL'83, European Computer Algebra Conference London, England, March 28–30, 1983 Proceedings*. Éd. par J. A. Hulzen. Springer Berlin Heidelberg, p. 146–156.

Lej84 Lejeune-Jalabert, Monique (1984). *Effectivité de calculs polynomiaux*. Vol. 19. Cours de l'Institut Fourier. Cours de DEA 84–85. Université de Grenoble I. url : http://www.numdam.org/numdam-bin/feuilleter?id=CIF_1984-1985__19_.

LL91 Lakshman, Y. N. et Daniel Lazard (1991). « On the complexity of zero-dimensional algebraic systems ». In : *Effective Methods in Algebraic Geometry*. Éd. par Teo Mora et Carlo Traverso. Boston, MA : Birkhäuser Boston, p. 217–225.

Mac94 Macaulay, F. S. (1994). *The algebraic theory of modular systems*. Cambridge Mathematical Library. Revised reprint of the 1916 original, With an introduction by Paul Roberts. Cambridge University Press.

MM82 Mayr, Ernst W. et Albert R. Meyer (1982). « The complexity of the word problems for commutative semigroups and polynomial ideals ». In : *Advances in Mathematics*, vol. 46, n°3, p. 305–329.

27. Normalisation de Noether

Résumé

Étant donné un système d'équations linéaires homogènes, les formules de Cramer permettent de paramétrer les solutions en fonction de certaines des variables. Le lemme de normalisation de Noether fournit un résultat analogue pour des systèmes d'équations polynomiales. Il permet d'amener le système dans une forme où les calculs se déroulent bien et ou les ensembles solutions de dimensions quelconques peuvent être représentés efficacement comme en dimension zéro.

27.1 Introduction

Nous commençons par examiner la situation des systèmes d'équations linéaires sous-déterminés. Le but de ce chapitre est d'étendre aux systèmes algébriques les concepts habituels, rappelés dans les paragraphes qui suivent. Soient $f_1,\dots,f_s$ des formes linéaires homogènes à $n+1$ variables $\mathrm{X}_0,\dots,\mathrm{X}_n$, à coefficients dans un corps k :

$$f_i(\mathrm{X}_0,\dots,\mathrm{X}_n)=\sum_{j=0}^{n} f_{ij}\mathrm{X}_j.$$

Point de vue de l'algèbre

On peut supposer, quitte à en enlever certaines, que les formes linéaires $f_1,\dots,f_s$ sont linéairement indépendantes (et donc forment une suite régulière). Comme la matrice F de dimension $s\times(n+1)$ qui leur est associée est de rang s, il existe un

mineur M non nul de taille s. Quitte à renuméroter les variables, F s'écrit donc

$$\mathrm{F} = \begin{array}{c} \begin{matrix} 0 \dots r & r+1 \dots n \end{matrix} \\ \begin{pmatrix} \mathrm{N} & \mathrm{M} \end{pmatrix} \end{array} \begin{matrix} 1 \\ \vdots \\ s \end{matrix}$$

avec $r = n - s$ et $\det \mathrm{M} \neq 0$. Alors nous avons l'équivalence

$$\mathrm{F}\begin{pmatrix} \mathrm{X}_0 \\ \vdots \\ \mathrm{X}_n \end{pmatrix} = 0 \iff \mathrm{M}\begin{pmatrix} \mathrm{X}_{r+1} \\ \vdots \\ \mathrm{X}_n \end{pmatrix} = -\mathrm{N}\begin{pmatrix} \mathrm{X}_0 \\ \vdots \\ \mathrm{X}_r \end{pmatrix} \iff \begin{pmatrix} \mathrm{X}_{r+1} \\ \vdots \\ \mathrm{X}_n \end{pmatrix} = -\mathrm{M}^{-1}\mathrm{N}\begin{pmatrix} \mathrm{X}_0 \\ \vdots \\ \mathrm{X}_r \end{pmatrix}.$$

Ainsi, les $n-r$ dernières coordonnées, que nous appellerons *variables liées*, sont données en fonction des $r+1$ premières, que nous appellerons *variables libres*. L'espace vectoriel des solutions devient ainsi le graphe d'une application linéaire de k^{r+1} dans k^s.

Point de vue de la géométrie

L'interprétation géométrique des formules de Cramer est la suivante : si le rang de la matrice F est s, nous pouvons partitionner l'ensemble des variables en deux paquets, $\mathrm{X}_0, \dots, \mathrm{X}_r$ et $\mathrm{X}_{r+1}, \dots, \mathrm{X}_n$ après renumérotation. Définissons les deux sous-espaces linéaires projectifs B d'équations $\mathrm{X}_{r+1} = \cdots = \mathrm{X}_n = 0$ et L d'équations $\mathrm{X}_0 = \cdots = \mathrm{X}_r = 0$. Le sous-espace linéaire d'équations $f_1 = \cdots = f_s = 0$ ne coupe pas L dans $\mathbb{P}^n$, et la projection de centre L l'envoie surjectivement sur B (cf. Définition 25.10).

Point de vue de l'algorithmique

À partir de la matrice F, l'algorithme du pivot de Gauss permet d'exhiber un mineur non nul de taille maximale, avec un nombre d'opérations arithmétiques polynomial en s et n. Ce calcul a été abordé au Chapitre 8.

27.2 Algorithme de mise en position de Noether

Notons $\mathrm{R} := k[\mathrm{X}_0, \dots, \mathrm{X}_n]$, et soient $f_1, \dots, f_s$ des polynômes homogènes de R de degrés respectifs $d_1, \dots, d_s$:

$$f_i(\mathrm{X}_0, \dots, \mathrm{X}_n) = \sum_{|a|=d_i} f_{i,a} \mathrm{X}_0^{a_0} \cdots \mathrm{X}_n^{a_n}$$

où $a = (a_0, \dots, a_n)$ représente un élément de $\mathbb{N}^{n+1}$ de degré $|a| = a_0 + \cdots + a_n$.

Point de vue de l'algèbre

Au chapitre précédent, un idéal I homogène de R a été défini comme en *position de Noether* (Définition 26.8) s'il existe $r \in \{0, \dots, n\}$ tel que $\mathrm{I} \cap k[\mathrm{X}_0, \dots, \mathrm{X}_r] = (0)$, et R/I est une extension entière de $k[\mathrm{X}_0, \dots, \mathrm{X}_r]$. Cette position partitionne les variables en variables dites *libres* $(\mathrm{X}_0, \dots, \mathrm{X}_r)$ et variables *liées* $(\mathrm{X}_{r+1}, \dots, \mathrm{X}_n)$. Lorsque k est infini, il

est toujours possible de s'y ramener, grâce à une méthode appelée normalisation de Noether, que nous allons décrire.

Si M est une matrice carrée de taille $n+1$ à coefficients dans k, et si $f \in k[X_0,\dots,X_n]$, on définit $f \circ M(X_0,\dots,X_n) := f(M \cdot {}^t(X_0,\dots,X_n))$. Si I est un idéal alors on note $I \circ M$ l'idéal engendré par les $f \circ M$ pour $f \in I$.

Proposition 27.1 — Version algébrique du lemme de normalisation de Noether. Soit I un idéal homogène de $R = k[X_0,\dots,X_n]$. Si k est infini, alors il existe un entier r et une matrice M inversible telle que $I \circ M$ soit en position de Noether.

Exemple 27.1 Avec $n = 2$, en changeant X_1 en $X_1 + X_2$ dans $I = (X_1X_2 - X_0^2)$, l'idéal devient $(X_2^2 + X_1X_2 - X_0^2)$ qui est bien en position de Noether, avec $r = 1$. La situation affine correspondante avait été illustrée en Exemple 25.10.

La preuve de la proposition repose sur un lemme technique.

Lemme 27.2 Soient $A \subseteq B$ une extension d'anneaux, et $b, c \in B$. Si b est entier sur A et c est entier sur l'anneau $A[b]$, alors c est entier sur A.

Démonstration. Écrivons les relations de dépendance entière sous la forme

$$P(b) = b^q + \sum_{i=0}^{q-1} a_i b^i = 0 \qquad \text{et} \qquad Q(c) = c^t + \sum_{j=0}^{t-1} A_j(b) c^j = 0,$$

avec $a_i \in A$ et $A_j \in A[X]$. Considérons ces relations comme des polynômes à coefficients dans $A[c]$ évalués en b. Leur résultant $\mathrm{Res}(P,Q)$ est nul car ils ont b comme racine commune. Dans le développement du déterminant de la matrice de Sylvester, la diagonale produit le terme de plus grand degré, en l'occurrence tq, en c. Ce résultant est donc une relation unitaire à coefficients dans A. ∎

En appliquant ce lemme à d entier sur $A[b,c]$ on obtient que d est entier sur A. Les cas particuliers $d = b + c$ et $d = bc$ montrent que la somme et le produit d'entiers sont entiers, et ainsi plus généralement les évaluations en des entiers sur A de polynômes à coefficients dans A.

Démonstration de la Proposition 27.1. La preuve construit la matrice M par produit de matrices carrées de dimension $n+1$ dont la forme par blocs est

$$M_i := \begin{pmatrix} B_i & 0 \\ 0 & I_{n-i} \end{pmatrix}, \tag{27.1}$$

le bloc B_i étant donné par

$$B_i := \begin{pmatrix} 1 & 0 & \cdots & 0 & a_{i,0} \\ 0 & 1 & \cdots & 0 & a_{i,1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & a_{i,i-1} \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \tag{27.2}$$

pour des coefficients $a_{i,0}, \ldots, a_{i,i-1}$ que nous allons choisir.

Soit $A_i := k[X_0, \ldots, X_i]$, et $J_i := I \circ M \cap A_i$. La preuve effectue une récurrence descendante sur i, à partir de $i = n$, en montrant qu'il existe une matrice M inversible, de la forme $M_n \cdots M_{i+1}$ qui rend $X_{i+1}, \ldots, X_n$ entiers sur A_i modulo $I \circ M$. La construction s'arrête avec le premier incide i rencontré tel que $J_i = (0)$.

Cette hypothèse est triviale avec $i = n$ (M étant la matrice identité). On suppose donc cette hypothèse vérifiée pour $i \leq n$. Si $J_i = (0)$, alors, par construction, M rend les variables $X_{i+1}, \ldots, X_n$ entières sur A_i et on trouve $r = i$. Cette construction donne bien une position de Noether.

Sinon, il existe un polynôme homogène f non nul dans J_i. La Proposition 25.2 donne alors un changement de variables ${}^t(X_0, \ldots, X_i) \mapsto B_i {}^t(X_0, \ldots, X_i)$ qui permet de supposer f unitaire en X_i. Après le changement de variables $M := M_n \cdots M_i$, X_i devient entière sur A_{i-1} modulo $I \circ M$, et les X_j pour $j > i$ sont inchangés, et sont donc entiers sur A_{i-1} modulo $I \circ M$ par le Lemme 27.2. L'hypothèse de récurrence est alors aussi bien satisfaite pour $i - 1$. ■

Point de vue de la géométrie

La version géométrique du lemme de normalisation de Noether dans le cadre projectif est la suivante :

Proposition 27.3 — Version géométrique du lemme de normalisation de Noether. Supposons k algébriquement clos. Pour tout idéal homogène I de dimension r, il existe deux sous-espaces linéaires L de codimension $r+1$ et B de dimension r telles que :
- $\mathbf{V}(I)$ et L ne se coupent pas,
- la restriction à $\mathbf{V}(I)$ de la projection π de centre L et de base B est surjective et finie, c'est-à-dire que pour tout point b de B, la fibre $\pi^{-1}(b)$ est constituée d'un nombre fini non nul de points.

Exemple 27.2 L'idéal $(X_1X_2 - X_0^2)$ des exemples précédents est une version homogénéisée de l'équation d'une hyperbole $X_1X_2 - 1 = 0$. La projection de l'hyperbole sur l'axe des X_1 d'équation $X_2 = 0$ rencontre tous les points sauf $X_1 = 0$, les fibres contiennent toutes exactement un point. Le changement de X_1 en $X_1 + X_2$ revient à considérer la droite $X_1 + X_2 = 0$ sur laquelle la projection devient surjective, et où chaque point a exactement un antécédent.

Démonstration. Par le Théorème 26.3, il existe un sous-espace linéaire L disjoint de

$\mathbf{V}(\mathrm{I})$ de codimension minimale égale à $r+1$. Soit B un espace de dimension r ne coupant pas L et soit π la projection de centre L et de base B.

La restriction de π à $\mathbf{V}(\mathrm{I})$ est surjective : si un point $b \in \mathrm{B}$ n'est pas une projection alors l'espace linéaire projectif $\mathrm{L}+b$ ne coupe pas $\mathbf{V}(\mathrm{I})$; cet espace est de codimension r ce qui contredit le Théorème 26.3.

Montrons maintenant que la fibre $\pi^{-1}(b) = \mathbf{V}(\mathrm{I}) \cap (\mathrm{L}+b)$ est finie, c'est-à-dire de dimension zéro. La variété $\mathrm{W} := \pi^{-1}(b) \subset \mathrm{L}+b$ n'intersecte par L, et comme L est de codimension 1 dans $\mathrm{L}+b$, le Théorème 26.3 implique que la dimension de W est au plus zéro. ■

La proposition suivante montre que la version algébrique de la normalisation de Noether implique la version géométrique. D'une manière indépendante du Théorème 26.5, on retrouve que r est la dimension de l'idéal, et qu'il est donc indépendant du procédé de calcul.

Proposition 27.4 Supposons k algébriquement clos. Soient I un idéal homogène de R et r un entier, tels que $\mathrm{I} \cap k[\mathrm{X}_0,\dots,\mathrm{X}_r] = (0)$ et R/I est une extension entière de $k[\mathrm{X}_0,\dots,\mathrm{X}_r]$. Soient $\mathrm{L} := \mathbf{V}(\mathrm{X}_0,\dots,\mathrm{X}_r)$, $\mathrm{B} := \mathbf{V}(\mathrm{X}_{r+1},\dots,\mathrm{X}_n)$, et π la projection de centre L et de base B. Alors $\mathbf{V}(\mathrm{I})$ n'intersecte pas L, la restriction de π à $\mathbf{V}(\mathrm{I})$ est surjective, et r est égal à la dimension de I.

Démonstration. Comme R/I est entier sur $k[\mathrm{X}_0,\dots,\mathrm{X}_r]$, pour tout $j \geq r+1$, il existe un entier m_j tel que $\mathrm{X}_j^{m_j} \in \mathrm{I} + (\mathrm{X}_0,\dots,\mathrm{X}_r)$. Par conséquent $\mathbf{V}(\mathrm{I})$ n'intersecte pas L. Comme $\mathrm{X}_0,\dots,\mathrm{X}_r$ forment une suite régulière, la dimension de L est $n-r-1$ par le Corollaire 26.8. Le Théorème 26.3 implique que I est de dimension au plus r.

Afin de montrer que π envoie $\mathbf{V}(\mathrm{I})$ surjectivement sur B nous pouvons nous restreindre à prouver que $\pi^{-1}(b)$ est non vide pour $b = (0 : \cdots : 0 : 1) \in \mathbb{P}^r$ (le 1 est la valeur de la coordonnée X_r), quitte à effectuer un changement de variables linéaire inversible à $\mathrm{X}_0,\dots,\mathrm{X}_r$. Soit J l'idéal de $\mathbf{V}(\mathrm{I}) \cap (\mathrm{L}+b)$, et soit f un polynôme de J. Notons f_j une relation entière pour l'image de X_j dans R/I, à coefficients dans $k[\mathrm{X}_0,\dots,\mathrm{X}_{j-1}]$, pour chaque $j \geq r+1$, et considérons

$$g := \mathrm{Res}_{\mathrm{X}_{r+1}}(f_{r+1}, \mathrm{Res}_{\mathrm{X}_{r+2}}(f_{r+2}, \mathrm{Res}_{\mathrm{X}_{r+3}}(\dots, \mathrm{Res}_{\mathrm{X}_n}(f_n, f)))).$$

En appliquant plusieurs fois la Proposition 6.7, on obtient $g \in \mathrm{J} \cap k[\mathrm{X}_0,\dots,\mathrm{X}_r]$, ce qui implique que $g(0,\dots,0,1,\mathrm{X}_{r+1},\dots,\mathrm{X}_n)$ est nul. Ensuite, en appliquant plusieurs fois la Proposition 6.8, on déduit que $f(0,\dots,0,1,\mathrm{X}_{r+1},\dots,\mathrm{X}_n)$ ne peut pas être constant. En particulier une puissance de X_r ne peut pas appartenir à J, et le Théorème 25.20 implique que $\mathbf{V}(\mathrm{J}) \neq \varnothing$. La restriction de π à $\mathbf{V}(\mathrm{I})$ est donc bien surjective et le Théorème 26.3 implique que I est de dimension au moins r. ■

Point de vue de l'algorithmique

En suivant la preuve de la Proposition 27.1 on obtient l'Algorithme 27.1, où $\mathcal{T}_{n+1}(k)$ représente l'ensemble des matrices carrées de taille $n+1$ sur k qui sont triangulaires supérieures avec des 1 sur la diagonale.

Entrée Un idéal I donné par un ensemble fini de générateurs homogènes.
Sortie Une matrice $M \in \mathcal{T}_{n+1}(k)$ telle que $I \circ M$ est en position de Noether, ainsi que la dimension r de I.

1. Initialiser i à n et M à la matrice identité.
2. Tant que $(I \circ M) \cap k[X_0, \dots, X_i] \neq (0)$ répéter :
   a. choisir $f \in (I \circ M) \cap k[X_0, \dots, X_i]$ non nul ;
   b. trouver $(a_{0,i}, \dots, a_{i-1,i}) \in k^i \setminus \{0\}$ qui n'annule pas f ;
   c. remplacer M par $M \cdot M_i$ (M_i définie par l'équation (27.1)) et i par $i-1$.
3. Renvoyer M et i.

Algorithme 27.1 – Algorithme de mise en position de Noether.

Proposition 27.5 Si k est infini, l'Algorithme 27.1 termine et le résultat renvoyé est correct.

Démonstration. L'algorithme est calqué sur la preuve de la Proposition 27.1. Si k est infini alors il termine et renvoie bien un changement de variables qui met l'idéal I en position de Noether. Le fait que l'entier i est égal à la dimension de I vient de la Proposition 27.4. ■

La partie calculatoire coûteuse de l'algorithme de mise en position de Noether est le test $(I \circ M) \cap k[X_0, \dots, X_i] \neq (0)$. Ce test peut être réalisé par un calcul de base standard en utilisant le Théorème 24.8, mais ceci peut s'avérer coûteux. Une autre façon d'aborder la mise en position de Noether est d'adopter un point de vue probabiliste, à partir de la proposition suivante :

Proposition 27.6 Soit I un idéal homogène de R. Il existe une variété affine propre W de $\mathcal{T}_{n+1}(k)$ telle que pour toute matrice $M \notin W$, l'idéal $I \circ M$ est en position de Noether.

Démonstration. Le changement de variables M renvoyé par l'Algorithme 27.1 est $M = M_n M_{n-1} \cdots M_{r+1}$, en reprenant les notations de la preuve de la Proposition 27.1. Quelles que soient les valeurs $a_{n,0}, \dots, a_{n,n-1}$ n'annulant pas un polynôme homogène non nul de I, les valeurs $a_{n-1,0}, \dots, a_{n-1,n-2}$ n'annulant pas un polynôme homogène non nul de $I \cap k[X_1, \dots, X_{n-1}]$, …, les valeurs $a_{r+1,0}, \dots, a_{r+1,r}$ n'annulant pas un polynôme homogène non nul de $I \cap k[X_1, \dots, X_{r+1}]$, l'idéal $I \circ M$ est en position de Noether. Ensuite, il suffit de remarquer que pour de telles valeurs et pour n'importe quelles valeurs de $a_{r,0}, \dots, a_{r,r-1}$, …, $a_{2,0}, a_{2,1}$, $a_{1,0}$ de k, l'idéal $I \circ M_n M_{n-1} \cdots M_1$ est toujours en position de Noether. ■

Corollaire 27.7 Soit I un idéal homogène de R. Il existe une variété affine propre W de $\mathcal{M}_{n+1}(k)$ telle que pour toute matrice $M \notin W$, M est inversible et l'idéal $I \circ M$

est en position de Noether.

Démonstration. L'ensemble des matrices carrées de taille $n+1$ ayant un mineur principal nul est contenu dans une variété propre de $\mathcal{M}_{n+1}(k)$, définie par l'annulation de l'un de ces mineurs principaux. Par ailleurs, il est classique que toute matrice M en dehors de cette dernière variété peut se décomposer en M = LU, où L est une matrice triangulaire inférieure et U est triangulaire supérieure avec des 1 sur la diagonale. Comme $I \circ L$ est en position de Noether si, et seulement si, I l'est, la conclusion provient de la proposition précédente. ■

Si k est infini, ou en tout cas suffisamment grand, on peut donc tirer au hasard un changement de variables qui met I en position de Noether avec une forte probabilité.

27.3 Utilisations de la position de Noether

Une application importante de la position de Noether est la représentation des variétés projectives de dimensions quelconques comme des variétés affines de dimension 0, pour lesquelles nous disposons de paramétrisations par éléments primitifs (Définition 25.4).

Notation

Dans ce chapitre et le suivant, nous allons souvent considérer un idéal homogène I de $R = k[X_0,\dots,X_n]$ de dimension r en position de Noether, et certaines de ses extensions. Pour bien distinguer les anneaux et corps utilisés dans les constructions, nous posons :

$$A_r := k[X_0,\dots,X_r], \quad B_r := R/I, \quad K_r := k(X_0,\dots,X_r), \quad E_r := K_r[X_{r+1},\dots,X_n]/I', \tag{27.3}$$

où I′ est l'extension de I dans $K_r[X_{r+1},\dots,X_n]$. Le quotient B_r ne dépend pas de r, mais cette notation rappelle qu'il s'agit d'une extension entière de A_r.

Réduction au cas affine de dimension 0

Informellement, il suffit d'étendre le corps de base k avec les variables libres. Cette construction a déjà été abordée dans la Section 26.4. du chapitre précédent.

Proposition 27.8 Soit I un idéal homogène de R de dimension $r \geq 0$ en position de Noether, et I′ l'extension de I dans $K_r[X_{r+1},\dots,X_n]$. Alors I′ est un idéal non homogène de dimension 0 (dans le cadre affine de la Définition 25.2). De plus, si I est radical alors I′ l'est aussi.

Démonstration. La première partie a déjà été vue au Théorème 26.5. Considérons un élément $g \in K_r[X_{r+1},\dots,X_n]$ et un entier $e \geq 1$ tels que g^e appartienne à I′. Par définition de I′, il existe $a \in A_r$ non nul tel que ag^e, et donc $(ag)^e$, soient dans I. Si I est radical, alors ag appartient lui aussi à I et donc g à I′. Cela prouve que I′ est aussi radical. ■

Exemple 27.3 On considère l'ideal (f_1, f_2) dans $R = k[X_0, X_1, X_2, X_3]$, avec

$$f_1 := X_1^2 + X_2^2 + X_3^2 - 2X_0^2, \quad f_2 := X_1^2 + X_2^2 - X_0^2.$$

L'idéal I' engendré par (f_1, f_2) dans $K_1[X_2, X_3]$ est de dimension 0, le quotient $E_1 = K_1[X_2, X_3]/I'$ admet $u = X_1 + 2X_2 - 3X_3$ comme élément primitif, et on obtient une paramétrisation de I' sous forme de Kronecker (Définition 25.5)

$$\begin{aligned} q(T) &= T^4 - 4X_1T^3 + (14X_1^2 - 26X_0^2)T^2 + \\ &\qquad (52X_0^2X_1 - 20X_1^3)T + 25X_0^4 + 14X_0^2X_1^2 + 25X_1^4, \\ w_2(T) &= 8(X_0^2 - X_1^2)(T^2 - 2X_1T + 5X_0^2 + 5X_1^2), \\ w_3(T) &= -12X_0^2(T^2 - 2X_1T - 5X_0^2 - 3X_1^2), \end{aligned}$$

de sorte que $I' = (q(u), q'(u)X_2 - w_2(u), q'(u)X_3 - w_3(u))$. Rappelons que le polynôme q n'est autre que le polynôme minimal de la multiplication par u modulo I'. Le reste de cette section montre comment relier une telle paramétrisation de I' à la variété $\mathbf{V}_{\bar{k}}(I)$.

Polynômes caractéristique et minimal

En Section 25.2, nous avons étudié les polynômes caractéristiques et minimaux des endomorphismes de multiplication pour construire les paramétrisations par élément primitif des idéaux radicaux de dimension 0. Nous allons reprendre cette approche en dimension quelconque dans le cadre projectif en exploitant la position de Noether.

Si f est un polynôme homogène de R, alors on note μ_f (resp. χ_f) le polynôme minimal (resp. caractéristique) de l'endomorphisme de multiplication par f dans E_r. Ces objets sont bien définis puisque la proposition précédente assure que E_r est une K_r-algèbre de dimension finie, dimension que nous noterons δ par la suite.

Exemple 27.4 Dans l'exemple précédent, on considère la multiplication par $f_3 = X_1 - 3X_2 + X_3$ dans $E_1 = \mathbb{Q}(X_0, X_1)[X_2, X_3]/(f_1, f_2)$. Une base standard de l'idéal engendré par les polynômes (f_1, f_2) dans $\mathbb{Q}(X_0, X_1)[X_2, X_3]$ pour l'ordre $\mathtt{tdeg}(X_3, X_2)$ (c'est-à-dire induit par $X_3 > X_2$) est donnée par $g_1 = f_2 = X_2^2 + X_1^2 - X_0^2$, $g_2 = f_1 - f_2 = X_3^2 - X_0^2$. Ceci fournit d'abord une base du quotient E_1 via les monômes sous l'escalier, à savoir $(1, X_2, X_3, X_2X_3)$. Ensuite, la matrice de multiplication par f_3 dans cette base vaut

$$\begin{pmatrix} X_1 & -3(X_0^2 - X_1^2) & X_0^2 & 0 \\ -3 & X_1 & 0 & X_0^2 \\ 1 & 0 & -3X_1 & -3(X_0^2 - X_1^2) \\ 0 & 1 & -3 & X_1 \end{pmatrix},$$

et son polynôme caractéristique est

$$\chi_{f_3}(T) = T^4 - 4X_1T^3 + 4(6X_1^2 - 5X_0^2)T^2 + 40X_1(X_0^2 - X_1^2)T + 4(X_0^2 - X_1^2)(16X_0^2 - 25X_1^2).$$

Ce polynôme est sans carré, il est donc aussi minimal.

A priori, les coefficients de ces polynômes χ_f et μ_f sont dans K_r, mais nous allons montrer que, comme dans cet exemple, il s'agit en fait de polynômes en les variables libres $X_0,\dots,X_r$, dont les degrés sont maîtrisés.

Définition 27.1 Un polynôme $q \in A_r[T]$ est dit *quasi-homogène* pour le poids d pour T et 1 pour $X_0,\dots,X_r$ si $q(X_0,\dots,X_r,T^d)$ est homogène. Le *quasi-degré* d'un tel polynôme q pour ces poids est le degré de $q(X_0,\dots,X_r,T^d)$.

Proposition 27.9 Soit I un idéal homogène de dimension $r \geq 0$ en position de Noether, et soit f un polynôme homogène de degré $d \geq 1$ de $k[X_0,\dots,X_n]$. Alors, le polynôme minimal $\mu_f(T)$ et le polynôme caractéristique $\chi_f(T)$ de l'endomorphisme de multiplication par f dans E_r appartiennent à $A_r[T]$, et sont quasi-homogènes si on associe le poids d à la variable T, et 1 aux variables $X_0,\dots,X_r$.

Démonstration. Puisque I est en position de Noether, f mod I est entier sur l'anneau A_r, et donc il existe un polynôme unitaire $q \in A_r[T]$ de degré m tel que $q(f) = 0$ dans E_r. Écrivons $q(f)$ sous la forme $\sum_{i=0}^{m} q_i f^i$, avec $q_i \in A_r$ et $q_m = 1$. Comme f est homogène, la composante homogène de degré dm de $q(f)$ s'écrit sous la forme $\sum_{i=0}^{m} q_{i,m-i} f^i$, où $q_{i,m-i}$ est la composante homogène de degré $m-di$ de q_i. Comme I est homogène, la Proposition 23.3 assure que le polynôme $\sum_{i=0}^{m} q_{i,m-i} f^i$ est dans I.

Par conséquent, le polynôme $Q(T) = \sum_{i=0}^{m} q_{i,m-i} T^i$ est une relation de dépendance entière pour f, qui est quasi-homogène. Par le lemme de Gauss (Théorème 21.2) tous les facteurs irréductibles de Q ont leurs coefficients dans A_r. Ils sont aussi quasi-homogènes : on peut écrire chaque facteur comme somme de ses composantes de quasi-degrés distincts, et dans le produit, seul demeure le produit des composantes de plus haut quasi-degré. Tout produit de puissances de ces facteurs irréductibles est quasi-homogène. C'est le cas de μ_f, qui est unitaire et qui divise Q (puisqu'il annule f), et de χ_f, qui a les mêmes facteurs irréductibles que μ_f. ■

Position de Noether simultanée

Pour développer des algorithmes incrémentaux dans la suite de cette partie, il est commode d'exiger une propriété un peu plus forte que la position de Noether.

Définition 27.2 Une suite régulière de polynômes homogènes $f_1,\dots,f_s$ de R est en *position de Noether simultanée* si chacun des *idéaux intermédiaires* $(f_1,\dots,f_i)$, pour $i \in \{1,\dots,s\}$, est lui-même en position de Noether.

Exemple 27.5 L'idéal $(f_1,f_2) := (X_2^2 + X_1^2 - X_0^2, X_1X_2 - X_0^2)$ de $R = k[X_0,X_1,X_2]$ est en position de Noether simultanée : f_1 est unitaire en X_2, et l'idéal contient le polynôme $X_1^2 f_1 - (X_0^2 + X_1X_2)f_2 = X_1^4 - X_1^2X_0^2 + X_0^4 \in k[X_0,X_1]$ qui est unitaire en X_1. En revanche l'idéal (f_2,f_1) n'est pas en position de Noether simultanée, comme le

montre l'Exemple 26.4.

Si un idéal I est engendré par une suite régulière de polynômes homogènes $f_1, \ldots, f_s$ de R, alors pour chaque valeur de i, le Corollaire 27.7 donne l'existence d'une variété propre W_i dans l'espace des changements de variables linéaires telle que tout changement de variables choisi en dehors de W_i soit inversible et mette l'idéal intermédiaire $I_i = (f_1, \ldots, f_i)$ en position de Noether. Donc quitte à appliquer un changement de variables linéaire pris à l'extérieur de $W_1 \cup \cdots \cup W_s$, à partir de maintenant nous pourrons librement supposer que $f_1, \ldots, f_s$ est en position de Noether simultanée.

Absence de torsion

La proposition précédente montre que les polynômes $\mu_f(f)$ et $\chi_f(f)$, qui appartiennent à I′ par le Lemme 25.8, sont des polynômes de R. Dans les prochains paragraphes nous montrons que l'inclusion $I \subseteq I' \cap R$ devient une égalité sous certaines « conditions de généricité », ce qui implique que $\mu_f(f)$ et $\chi_f(f)$ appartiennent aussi à I. Cette propriété est la clé pour montrer qu'une paramétrisation par élément primitif de I′ constitue une bonne représentation de $\mathbf{V}_{\bar{k}}(I)$.

Exemple 27.6 L'idéal $I = (X_2) \cap (X_2, X_1)^2$ de $R = A_2 = k[X_0, X_1, X_2]$ est en position de Noether, de dimension 1, et on vérifie que I′ vaut (X_2). Par conséquent l'inclusion $I \subseteq I' \cap R$ est en général stricte.

Si b est un élément de $I' \cap R$, alors il existe a non nul dans A_r tel que ab appartienne à I. Ainsi, pour prouver l'égalité $I = I' \cap R$, la définition suivante sera utile, appliquée au A_r-module $B_r = R/I$:

Définition 27.3 Soit A un anneau intègre, et soit M un A-module. Un élément $b \in M$ est de *torsion* s'il existe un élément $a \in A$ non nul tel que $ab = 0$. Le module M est dit *sans torsion* si zéro est son seul élément de torsion.

Exemple 27.7 Considérons $n = 2$, et $I = (X_2) \cap (X_1, X_2)^2$. L'idéal I est de dimension 1 et en position de Noether. L'image de X_2 dans $B_1 = R/I$ est non nulle, alors que celle de X_1X_2 l'est. Le A_1-module B_1 a donc de la torsion. En termes géométriques, $\mathbf{V}_{\bar{k}}(I)$ est une droite d'idéal annulateur (X_2). C'est le point $\mathbf{V}_{\bar{k}}((X_1, X_2)^2)$, que l'on peut considérer comme « multiple » sur la droite, qui est responsable de la torsion ici.

Exemple 27.8 Considérons $n = 2$, et $I = (X_2) \cap (X_2 - X_0, X_1)$. L'idéal I est de dimension 1 et en position de Noether. L'image de X_2 dans $B_1 = R/I$ est non nulle, alors que celle de X_1X_2 l'est. Le A_1-module B_1 a donc de la torsion. En termes géométriques, $\mathbf{V}_{\bar{k}}(I)$ est l'union d'une droite et d'un point, et c'est ce point qui est responsable de la torsion ici.

Nous allons maintenant prouver le résultat suivant, qui donne des conditions suffisantes pour que B_r soit sans torsion. Les notations sont celles de l'Équation (27.3).

Théorème 27.10 Soit I un idéal de R, homogène, radical et engendré par une suite régulière $f_1,\dots,f_{n-r}$ en position de Noether simultanée. Alors B_r est un A_r-module sans torsion. En particulier, on a $I = I' \cap R$, où I' est l'extension de I dans $K_r[X_{r+1},\dots,X_n]$.

La preuve utilise un raisonnement par récurrence sur le nombre d'éléments de la suite régulière. Un des arguments de l'étape de récurrence est fourni par le lemme suivant qui permet d'exhiber un polynôme unitaire en la nouvelle variable liée.

Lemme 27.11 Soit I un idéal homogène radical de dimension $r > 0$ en position de Noether, et soit f un polynôme homogène de R qui n'est pas diviseur de zéro dans B_r. Si B_r est un A_r-module sans torsion, et si $I+(f)$ est en position de Noether, alors le polynôme $\mu_f(0)$ est non nul et unitaire en X_r.

Démonstration. D'après le Théorème 26.7, la dimension de $I+(f)$ est $r-1$. Comme I et $I+(f)$ sont supposés en position de Noether, il existe un polynôme homogène non nul $a \in (I+(f)) \cap A_r$ qui est unitaire en X_r. Ensuite, il existe un polynôme homogène $g \in R$ tel que a s'écrive gf dans B_r. Écrivons le polynôme minimal de f dans E_r sous la forme

$$\mu_f(T) = T^m + \mu_{f,m-1}T^{m-1} + \cdots + \mu_{f,0}.$$

L'absence de torsion de B_r implique $\mu_f(f) \in I$. Si $\mu_{f,0} = \mu_f(0)$ était nul, alors on aurait $v(f)f \in I$, avec $v(T) = \mu_f(T)/T$. Comme f est non diviseur de zéro dans E_r, le polynôme $v(f)$ appartiendrait à I et serait un annulateur de f dans E_r qui divise strictement μ_f, ce qui est impossible de par la minimalité de μ_f.

Les polynômes a et $\mu_{f,0}$ étant non nuls dans K_r, le polynôme minimal de g dans E_r n'est autre que $T^m\mu_f(a/T)/\mu_{f,0}$, qui s'écrit

$$\mu_g(T) = T^m + \cdots + \frac{\mu_{f,m-1}a^{m-1}}{\mu_{f,0}}T + \frac{a^m}{\mu_{f,0}}.$$

Par la Proposition 27.9, les coefficients de μ_g sont dans A_r, et en particulier son coefficient constant. Par conséquent $\mu_{f,0}$ divise a^m dans $A_{r-1}[X_r]$. Ceci montre bien que $\mu_f(0)$ est unitaire en X_r. ∎

Le raisonnement sur l'ajout d'un nouveau polynôme f passe par une étape intermédiaire :

Lemme 27.12 Soit I un idéal homogène radical de dimension $r > 0$ en position de Noether tel que B_r est un A_r-module sans torsion, et soit f un polynôme homogène de R qui n'est pas diviseur de zéro dans B_r et tel que $I+(f)$ est en position de Noether. En posant $R_Y = A_r[Y, X_{r+1},\dots,X_n]$ et $I_Y = I+(Y-f)$, on a les propriétés suivantes :

i. $I_Y \cap A_{r-1}[Y] = (0)$;
ii. R_Y/I_Y est une extension entière de l'anneau $A_{r-1}[Y]$;

iii. Le polynôme minimal ρ_g de la multiplication par un polynôme g de R dans

$$E_Y := k(X_0,\dots,X_{r-1},Y)[X_{r+1},\dots,X_n]/I'_Y$$

a ses coefficients dans $A_{r-1}[Y]$, où I'_Y représente l'extension de I_Y dans

$$k(X_0,\dots,X_{r-1},Y)[X_{r+1},\dots,X_n];$$

iv. R_Y/I_Y est un $A_{r-1}[Y]$-module sans torsion.

Démonstration. Supposons qu'il existe un polynôme P non nul dans $A_{r-1}[T]$ tel que $P(f) \in I$, et choisissons P de plus petit degré possible en T avec cette propriété. On obtient alors $P(0) \in I + (f)$. La position de Noether de $I + (f)$ implique $P(0) = 0$. En posant $\hat{P}(T) = P(T)/T$, on déduit $\hat{P}(f)f \in I$, puis $\hat{P}(f) \in I$ puisque f est non diviseur de 0 dans B_r. Ceci conduit à une contradiction avec le choix de P de degré minimal en T, et achève la preuve de l'assertion (i).

La Proposition 27.9 et le Lemme 27.11 assurent que μ_f définit une relation entière pour X_r sur $A_{r-1}[Y]$ modulo I_Y. Le Lemme 27.2 implique alors l'assertion (ii).

L'idéal I_Y n'étant pas homogène en général, l'assertion (iii) ne découle pas de la Proposition 27.9. Il faut étendre légèrement les arguments utilisés. D'une part la première partie de la preuve du Théorème 26.5 s'adapte aisément pour montrer que E_Y est une $k(X_0,\dots,X_{r-1},Y)$-algèbre de dimension finie. Le polynôme minimal ρ_g est ainsi bien défini.

Nous adaptons enfin le début de la preuve de la Proposition 27.9. Il existe un polynôme unitaire $Q \in A_{r-1}[Y][T]$ tel que $Q(g) \in I_Y$. Par le lemme de Gauss (Théorème 21.2) tous les facteurs irréductibles de Q ont leurs coefficients dans $A_{r-1}[Y]$ et sont unitaires. Comme ρ_g divise Q dans $k(X_0,\dots,X_{r-1},Y)[T]$ et qu'il est unitaire, ρ_g a forcément ses coefficients dans $A_{r-1}[Y]$. Cela achève la preuve de (iii). Mentionnons que nous n'aurons pas besoin de borner les degrés des coefficients de ρ_g dans la suite.

Examinons la torsion de R_Y/I_Y. Soient a un polynôme non nul de $A_{r-1}[Y]$, et g un polynôme de $A_{r-1}[Y,X_{r+1},\dots,X_n]$ tels que ag soit dans I_Y. On examine le résultant en Y suivant :

$$b(X_0,\dots,X_r) = \mathrm{Res}_Y(a(X_0,\dots,X_{r-1},Y),\mu_f(X_0,\dots,X_r,Y)).$$

D'une part b appartient à A_r, et d'autre part la Proposition 6.7 assure l'existence de polynômes $U(X_0,\dots,X_r,Y)$ et $V(X_0,\dots,X_r,Y)$ tels qu'on ait la relation de Bézout

$$b(X_0,\dots,X_r) = U(X_0,\dots,X_r,Y)a(X_0,\dots,X_{r-1},Y) + V(X_0,\dots,X_r,Y)\mu_f(X_0,\dots,X_r,Y).$$

Rappelons que b est le déterminant de la matrice de Sylvester (Section 6.2) de a et μ_f vus comme polynômes en la variable Y. Comme a ne dépend pas de la variable X_r, comme μ_f est unitaire, homogène, et de coefficient constant $\mu_f(0)$ unitaire en X_r, un simple examen de cette matrice montre que le coefficient de plus haut degré en X_r est proportionnel sur k à $\mathrm{ct}_Y(a)^{\deg \mu_f}$ (où $\mathrm{ct}_Y(a)$ est le coefficient de tête de a vu dans $A_{r-1}[Y]$). Par conséquent b est non nul. En combinant $ag \in I_Y$ et $\mu_f(X_0,\dots,X_r,Y) \in I_Y$ avec la relation de Bézout, on obtient $bg \in I_Y$. En substituant f à Y et en utilisant l'absence de torsion de B_r, on déduit que $g(X_0,\dots,X_{r-1},f)$ appartient à I et donc que g appartient à I_Y. ■

Nous arrivons maintenant à l'énoncé clé suivant, qui passe d'un module sans torsion à un suivant, obtenu en ajoutant un non diviseur de zéro à l'idéal :

Proposition 27.13 Soit I un idéal homogène radical de dimension $r > 0$ en position de Noether, et soit f un polynôme homogène de R qui n'est pas diviseur de zéro dans B_r et tel que $I + (f)$ est en position de Noether. Si B_r est un A_r-module sans torsion, alors le A_{r-1}-module $R/\sqrt{I + (f)}$ est sans torsion.

Démonstration. On a déjà vu dans la preuve du Lemme 27.11, que $I + (f)$ est de dimension $r - 1$. Soient $b \in R$ et $a \in k[X_0, \dots, X_{r-1}] \setminus \{0\}$ tels que ab appartienne à $\sqrt{I + (f)}$. On veut montrer que b appartient à $\sqrt{I + (f)}$. Il existe un entier e tel que $(ab)^e$ appartienne à $I + (f)$. On note g un polynôme tel que $a^e b^e - gf$ appartient à I.

On utilise maintenant le lemme précédent, avec les mêmes notations. Les polynômes minimaux des endomorphismes de multiplication par g et par b^e dans E_Y, notés respectivement $\rho_g(T)$ et $\rho_{b^e}(T)$, appartiennent à $A_{r-1}[Y][T]$. Écrivons $\rho_g(T) = T^m + \rho_{g,m-1} T^{m-1} + \cdots + \rho_{g,0}$. Comme $a^e b^e - gY$ appartient à I_Y, et que a^e et Y sont dans $A_{r-1}[Y]$, le polynôme minimal $\rho_{b^e}(T)$ est

$$\rho_{b^e}(T) = \left(\frac{Y}{a^e}\right)^m \rho_g\left(\frac{a^e T}{Y}\right) = T^m + \rho_{g,m-1}\frac{Y}{a^e} T^{m-1} + \cdots + \rho_{g,0}\left(\frac{Y}{a^e}\right)^m.$$

Par conséquent a^{je} divise $\rho_{g,m-j} Y^j$ et donc $\rho_{g,m-j}$ dans $k[X_0, \dots, X_{r-1}, Y]$, pour tout $j \in \{0, \dots, m-1\}$. L'assertion (iv) du lemme précédent assure que $\rho_{b^e}(b^e)$ appartient à I_Y. En substituant Y par f, on en déduit que b^{em} appartient à $I + (f)$, et ainsi $b \in \sqrt{I + (f)}$. ∎

Nous pouvons enfin conclure.

Fin de la démonstration du Théorème 27.10. On commence par appliquer la proposition précédente avec $I = R$ et $f = f_1$. On obtient que $R/\sqrt{(f_1)}$ est un A_{n-1}-module sans torsion. On utilise ensuite à nouveau la proposition précédente avec $I = \sqrt{(f_1)}$ et $f = f_2$. Il s'ensuit que $R/\sqrt{\sqrt{(f_1)} + (f_2)}$ est un A_{n-2}-module sans torsion. Il est aisé de vérifier que $\sqrt{\sqrt{(f_1)} + (f_2)}$ coïncide avec $\sqrt{(f_1, f_2)}$. On continue de la même manière à prouver par récurrence que $R/\sqrt{(f_1, \dots, f_i)}$ n'a pas de A_{n-i}-torsion, jusqu'à $i = n - r$. ∎

Corollaire 27.14 Soit I un idéal homogène radical engendré par une suite régulière $f_1, \dots, f_{n-r}$ en position de Noether simultanée, et soit f un polynôme homogène non nul. Alors, le polynôme caractéristique χ_f et minimal μ_f de f dans E_r satisfont :

i. $\mu_f(f) = \chi_f(f) = 0$ dans B_r ;
ii. $\mu_f(0) = 0$ si et seulement si $\chi_f(0) = 0$, si et seulement si f divise zéro dans B_r.

Démonstration. La première assertion est une conséquence de la Proposition 27.9 et du Théorème 27.10. Pour la deuxième assertion, si $\mu_f(0) = 0$, alors on a $\nu_f(f)f = 0$ avec $\nu_f(T) = \mu_f(T)/T$ et $\nu_f(f) \neq 0$ dans B_r (par minimalité de μ_f). Par conséquent, f est diviseur de zéro. Réciproquement, si f est diviseur de zéro, alors il existe $g \notin I$ tel que $fg \in I$. De $g\mu_f(f) \in I$ on déduit $\mu_f(0)g \in I$. L'absence de torsion implique $\mu_f(0) = 0$. Finalement, comme χ_f et μ_f ont les mêmes facteurs irréductibles, l'annulation de $\chi_f(0)$ est équivalente à celle de $\mu_f(0)$. ■

Exemple 27.9 On considère l'idéal $I = (f_1, f_2)$ de l'Exemple 27.3. Comme f_1 est non nul, il n'est pas diviseur de zéro dans R. De plus, (f_1) est bien en position de Noether puisque f_1 est unitaire en X_3. Une base de $R/(f_1)$ est $\{1, X_3\}$. Dans cette base, on peut calculer la matrice de multiplication par f_2 et obtenir ainsi le coefficient constant de son polynôme caractéristique :

$$\chi_{f_2}(0) = X_2^4 + 2X_1^2X_2^2 - 2X_0^2X_2^2 + X_1^4 - 2X_0^2X_1^2 + X_0^4.$$

Par le corollaire précédent, on en déduit d'abord que f_2 n'est pas diviseur de zéro dans $R/(f_1)$. Ensuite, comme $\chi_{f_2}(0)$ est unitaire en X_2 à coefficients dans $k[X_0, X_1]$ et appartient à I, il s'ensuit que I en position de Noether.

R Supposer I engendré par une suite régulière n'est pas une hypothèse habituelle. Il est plus naturel de considérer des idéaux I de *dimension pure* (*unmixed* en anglais), ce qui signifie que tous les *premiers associés* de I sont de même dimension. Ces considérations dépassent le cadre de cette partie, dont l'objectif se limite principalement à établir une preuve complète de l'algorithme de résolution géométrique du chapitre suivant.

27.4 Paramétrisation en dimension quelconque

Nous continuons de supposer I radical de dimension r engendré par une suite régulière, et conservons les notations de la section précédente. Les calculs de polynômes minimaux servent à obtenir des paramétrisations par élément primitif de I'. Si k est de caractéristique nulle ou bien $> \delta = \dim_{K_r} E_r$, alors on peut utiliser la Proposition 25.10 : pour tout $(\lambda_0 : \cdots : \lambda_n)$ à l'extérieur d'une variété projective propre de $\mathbb{P}^n$, la forme linéaire $u = \lambda_{r+1}X_{r+1} + \cdots + \lambda_nX_n$ est primitive pour I'. Le polynôme minimal μ_u de u dans E_r est homogène de degré total δ. Nous montrons maintenant que les autres polynômes de la paramétrisation sous forme de Kronecker (Définition 25.5) ont aussi degré δ.

Revenons à la construction des paramétrisations en dimension 0 de la Section 25.2. Il nous faut introduire les variables $\Lambda_{r+1}, \ldots, \Lambda_n$, et les notations suivantes :

$$\begin{aligned} k_\Lambda &= k(\Lambda_{r+1}, \ldots, \Lambda_n), & R_\Lambda &= k_\Lambda[X_0, \ldots, X_n], \\ A_{\Lambda,r} &= k_\Lambda[X_0, \ldots, X_r], & K_{\Lambda,r} &= k_\Lambda(X_0, \ldots, X_r), \\ B_{\Lambda,r} &= R_{\Lambda,r}/I_\Lambda, & E_{\Lambda,r} &= K_{\Lambda,r}[X_{r+1}, \ldots, X_n]/I'_\Lambda, \end{aligned}$$

où I_Λ (resp. I'_Λ) représente l'extension de I dans R_Λ (resp. dans $K_{\Lambda,r}[X_{r+1},\dots,X_n]$). En fait cette extension avec les Λ_i ne change pas grand chose à la situation, comme précisé dans le lemme suivant :

Lemme 27.15 Soit I un idéal homogène radical engendré par une suite régulière $f_1,\dots,f_{n-r}$ en position de Noether simultanée. On suppose k de caractéristique nulle ou bien $> \delta = \dim_{K_r} E_r$. Alors on a
- la suite $f_1,\dots,f_n$ est régulière dans R_Λ ;
- I_Λ est radical ;
- $\dim I_\Lambda = \dim I = r$, et $\deg I_\Lambda = \deg I$.

De plus, le polynôme minimal $\mu_{u_\Lambda}(T)$ de $u_\Lambda = \Lambda_{r+1}X_{r+1} + \cdots + \Lambda_n X_n$ dans $E_{\Lambda,r}$ satisfait les propriétés suivantes :
- $\mu_{u_\Lambda}(T)$ est un polynôme homogène sans carré de $K_\Lambda[X_0,\dots,X_{r-1},T]$ de degré total δ ;
- $\mu_{u_\Lambda}(T)$ appartient à $k[\Lambda_{r+1},\dots,\Lambda_n,X_0,\dots,X_r,T]$, et $\mu_{u_\Lambda}(u_\Lambda)$ appartient à l'extension I^Λ de I à $R^\Lambda = k[\Lambda_{r+1},\dots,\Lambda_n,X_0,\dots,X_n]$.

Démonstration. Le fait que la suite $f_1,\dots,f_r$ est régulière dans R_Λ provient du Corollaire 26.12. Puis le Lemme 25.14 assure que I_Λ est radical. Pour la dimension et le degré de I_Λ, on peut soit utiliser le Corollaire 26.8, ou bien constater que I et I_Λ ont mêmes bases standard (les calculs de base standard ne dépendent que du corps engendré par les coefficients des polynômes en entrée).

La Proposition 27.9 utilisée avec K_Λ assure que μ_{u_Λ} est homogène en $X_0,\dots,X_r,T$, sans carré, et de degré total δ. Pour la dernière assertion, on utilise le Corollaire 27.14, en voyant $f_1,\dots,f_{n-r}$ comme une suite régulière engendrant I^Λ : en effet il est aisé de vérifier que la position de Noether simultanée est conservée, et que I^Λ est radical de dimension n. ■

Théorème 27.16 Soit I un idéal homogène radical engendré par une suite régulière $f_1,\dots,f_{n-r}$ en position de Noether simultanée. On suppose k de caractéristique nulle ou bien $> \delta = \dim_{K_r} E_r$. Soit $u = \lambda_{r+1}X_{r+1} + \cdots + \lambda_n X_n$ un élément primitif pour l'extension I′ de I dans $K_r[X_{r+1},\dots,X_n]$, et soit $q = \mu_u, w_{r+1},\dots,w_n$ les polynômes de la paramétrisation associée sous forme de Kronecker :

$$I' = (q(u), q'(u)X_{r+1} - w_{r+1}(u),\dots,q'(u)X_n - w_n(u)).$$

Alors, les polynômes $q, w_{r+1},\dots,w_n$ sont homogènes de degrés totaux δ. De plus, on a l'inclusion

$$(q(u), q'(u)X_{r+1} - w_{r+1}(u),\dots,q'(u)X_n - w_n(u)) \subseteq I. \tag{27.4}$$

Démonstration. Tout d'abord, d'après le Corollaire 25.16, le polynôme minimal $\mu_u = q$ est la spécialisation en $\Lambda_{r+1} = \lambda_{r+1}, \dots, \Lambda_n = \lambda_n$ du polynôme minimal μ_{u_Λ} de $u_\Lambda = \Lambda_{r+1}X_{r+1} + \cdots + \Lambda_n X_n$ de l'extension I'_Λ. Aussi, chaque polynôme w_i est la spécialisation de $-\partial\mu_{u_\Lambda}/\partial\Lambda_i$. Le lemme précédent assure ainsi que $q, w_{r+1},\dots,w_n$ sont homogènes

de degrés totaux δ. De plus ce lemme permet d'écrire $\mu_{u_\Lambda}(u_\Lambda)$ sous la forme

$$\mu_{u_\Lambda}(u_\Lambda) = g_1 f_1 + \cdots + g_{n-r} f_{n-r},$$

avec des g_i dans R^Λ. Par conséquent $\frac{\partial \mu_{u_\Lambda}}{\partial \mathrm{T}}(u_\Lambda)\mathrm{X}_i + \frac{\partial \mu_{u_\Lambda}}{\partial \Lambda_i}(u_\Lambda)$ appartient aussi à I^Λ, pour tout i de $r+1$ à n. On peut enfin spécialiser les variables Λ_i sans risque, pour obtenir la dernière inclusion du théorème. ■

Exemple 27.10 En général l'inclusion des idéaux (27.4) du théorème précédent est stricte, comme le montre l'exemple suivant. On prend $n = 2$,

$$f_1 = \mathrm{X}_2^2 + \mathrm{X}_1\mathrm{X}_2 + \mathrm{X}_0^2, \quad f_2 = \mathrm{X}_0\mathrm{X}_2 - \mathrm{X}_1^2 + \mathrm{X}_0\mathrm{X}_1.$$

Pour l'ordre 1ex induit par $\mathrm{X}_0 < \mathrm{X}_1 < \mathrm{X}_2$, une base standard est formée de f_1, f_2 et

$$\mathrm{X}_1^4 - \mathrm{X}_0\mathrm{X}_1^3 + \mathrm{X}_0^4, \quad \mathrm{X}_1^2\mathrm{X}_2 + \mathrm{X}_0^3.$$

Les monômes de tête sont donc $\{\mathrm{X}_2^2, \mathrm{X}_0\mathrm{X}_2, \mathrm{X}_1^2\mathrm{X}_2, \mathrm{X}_1^4\}$. On déduit de cette base la paramétrisation sous forme de Kronecker pour l'élément primitif X_1 :

$$q(\mathrm{T}) = \mathrm{T}^4 - \mathrm{X}_0\mathrm{T}^3 + \mathrm{X}_0^4, \quad w_1(\mathrm{T}) = q'(\mathrm{T})\mathrm{T} \bmod q(\mathrm{T}) = \mathrm{X}_0\mathrm{T}^3 - 4\mathrm{X}_0^4,$$
$$w_2(\mathrm{T}) = -q'(\mathrm{T})\mathrm{T}^{-2}\mathrm{X}_0^3 \bmod q(\mathrm{T}) = -4\mathrm{X}_0^3\mathrm{T} + 3\mathrm{X}_0^4.$$

Un nouveau calcul de base standard, pour l'idéal engendré par $q(\mathrm{X}_1)$, $q'(\mathrm{X}_1)\mathrm{X}_1 - w_1(\mathrm{X}_1)$ et $q'(\mathrm{X}_1)\mathrm{X}_2 - w_2(\mathrm{X}_1)$, donne les monômes de tête assez différents

$$\{\mathrm{X}_1^4, \mathrm{X}_1^3\mathrm{X}_2, \mathrm{X}_0^2\mathrm{X}_1^2\mathrm{X}_2, \mathrm{X}_0^4\mathrm{X}_1\mathrm{X}_2, \mathrm{X}_0^6\mathrm{X}_2\}.$$

On en déduit par exemple que le polynôme f_2 n'est pas dans l'idéal.

Interprétation géométrique

Revenons à l'interprétation géométrique de la position de Noether de la Proposition 27.4. Soit I un idéal homogène radical de dimension r. On note π la projection de centre $\mathbf{V}_{\bar{k}}(\mathrm{X}_0, \ldots, \mathrm{X}_r)$ et de base $\mathbf{V}_{\bar{k}}(\mathrm{X}_{r+1}, \ldots, \mathrm{X}_n)$. Alors la restriction de π à $\mathbf{V}_{\bar{k}}(\mathrm{I})$ est bien définie et surjective. La question que nous abordons ici est le calcul d'une paramétrisation des points de $\pi^{-1}(a)$ pour un point a donné.

Proposition 27.17 Soit I un idéal homogène radical engendré par une suite régulière $f_1, \ldots, f_{n-r}$ en position de Noether simultanée. On suppose k de caractéristique nulle ou bien $> \delta = \dim_{\mathrm{K}_r} \mathrm{E}_r$. Considérons une paramétrisation sous forme de Kronecker $q, w_{r+1}, \ldots, w_n$ de I' pour un élément primitif $u = \lambda_{r+1}\mathrm{X}_{r+1} + \cdots + \lambda_n\mathrm{X}_n$. Soit $(a_0 : \cdots : a_r)$ un point de $\mathbb{P}^r$, et notons

$$\mathrm{Q}(\mathrm{T}) := q(a_0, \ldots, a_r, \mathrm{T}), \mathrm{W}_{r+1}(\mathrm{T}) := w_{r+1}(a_0, \ldots, a_r, \mathrm{T}), \ldots, \mathrm{W}_n(\mathrm{T}) := w_n(a_0, \ldots, a_r, \mathrm{T}).$$

Alors, il existe une variété projective propre Z de $\mathbb{P}^r$ telle que pour tout choix de $(a_0 : \cdots : a_r)$ à l'extérieur de Z on a les propriétés suivantes :
i. $I_a = I + (X_0 - a_0, \ldots, X_r - a_r)$ est un idéal non homogène radical de dimension 0 ;
ii. $Q, W_{r+1}, \ldots, W_n$ forment une paramétrisation sous forme de Kronecker par l'élément primitif u pour $I_a \cap k[X_{r+1}, \ldots, X_n]$.

Démonstration. D'abord, on utilise l'inclusion des idéaux (27.4) du Théorème 27.16. Pour toute spécialisation $X_0 = a_0, \ldots, X_r = a_r$, pour laquelle Q est sans carré, on a

$$(Q(u), Q'(u)X_{r+1} - W_{r+1}(u), \ldots, Q'(u)X_n - W_n(u)) \subseteq I_a,$$

ce qui implique que les puissances de u forment une famille génératrice de R/I_a, et que I_a est un idéal non homogène radical de dimension 0.

Ensuite, on considère l'inclusion dans l'autre sens. Pour tout i de 1 à $n-r$, il existe des polynômes $h_{i,j}$ homogènes dans R, et un polynôme homogène non nul b dans A_r tels que

$$b f_i = h_{i,0} q(u) + \sum_{j=r+1}^{n} (q'(u)X_j - w_j(u)) h_{i,j}.$$

Pour toute spécialisation $X_0 = a_0, \ldots, X_r = a_r$ n'annulant pas b on a donc l'inclusion en sens inverse

$$(Q(u), Q'(u)X_{r+1} - W_{r+1}(u), \ldots, Q'(u)X_n - W_n(u)) \supseteq I_a.$$

Finalement, en notant c le discriminant de q en T (qui est homogène) on peut prendre $\mathbf{V}_{\bar{k}}(b) \cup \mathbf{V}_{\bar{k}}(c)$ pour Z. ■

En dimension 0, on obtient le résultat intuitif suivant, disant que les situations affines et projectives de dimension 0 sont les mêmes.

Corollaire 27.18 Soit I un idéal homogène radical engendré par une suite régulière $f_1, \ldots, f_n$ en position de Noether simultanée. On suppose k de caractéristique nulle ou bien $> \delta = \dim_{K_0} E_0$. Considérons une paramétrisation homogène $q, v_1, \ldots, v_n$ de I′ pour un élément primitif $u = \lambda_1 X_1 + \cdots + \lambda_n X_n$. Alors $\mathbf{V}_{\bar{k}}(I)$ n'a aucun point à l'infini, et la spécialisation de la paramétrisation en $X_0 = 1$ est une paramétrisation de l'idéal non homogène $(I + (X_0 - 1)) \cap k[X_1, \ldots, X_n]$.

Démonstration. L'absence de point à l'infini découle juste de la Proposition 27.4. Le reste est une conséquence de la proposition précédente. ■

À ce stade, étant donnée une paramétrisation de I′, la Proposition 27.17 fournit une paramétrisation de $\pi^{-1}(a_0, \ldots, a_r)$ pour presque tout point $(a_0 : \cdots : a_r)$ de $\mathbb{P}^r$. Pour les autres points, il est aussi possible de décrire $\pi^{-1}(a_0, \ldots, a_r)$ en utilisant le théorème suivant.

Théorème 27.19 Soit I un idéal homogène radical engendré par une suite régulière $f_1, \ldots, f_{n-r}$ en position de Noether simultanée. On suppose k de caractéristique nulle ou bien $> \delta = \dim_{K_r} E_r$. Considérons donnée une paramétrisation

$q, w_{r+1}, \dots, w_n$ de I′ pour un élément primitif $u = \lambda_{r+1}X_{r+1} + \cdots + \lambda_n X_n$. Alors, pour tout point $(a_0 : \cdots : a_r)$ de $\mathbb{P}^r$ tel que u est primitif pour l'idéal non homogène $I_a = \sqrt{I + (X_0 - a_0, \dots, X_r - a_r)}$, on note $\tilde{Q}(T) = q(a_0, \dots, a_r, T)$ et $\tilde{W}_i(T) = w_i(a_0, \dots, a_r, T)$, et on obtient :

— $P = \gcd(\tilde{Q}, \tilde{Q}')$ divise chaque $\tilde{W}_i(T)$,

— $Q = \tilde{Q}/\tilde{P}$, $W_i = (\tilde{W}_i/P)(\tilde{Q}'/P)^{-1} \bmod Q$ pour i de $r+1$ à n forment une paramétrisation de $I_a \cap k[X_{r+1}, \dots, X_n]$.

Nous ne donnons pas la preuve de ce résultat, qui n'est pas utilisé par la suite, et renvoyons le lecteur aux notes ci-dessous pour une référence.

Notes

Le point de vue utilisé dans ce chapitre pour la normalisation de Noether provient essentiellement de l'article de Giusti de 1988, qui décrit un algorithme de complexité polynomiale en d^{n^2}, où d est le maximum des degrés des générateurs de l'idéal [Giu88]. Cet algorithme a ensuite été amélioré en collaboration avec Heintz [GH93] pour obtenir une complexité non uniforme polynomiale en d^n.

Lorsque le cardinal de k est infini, ou en tout cas suffisamment grand, Giusti, Hägele, Lecerf, Marchand et Salvy ont implanté cet algorithme de normalisation, et ont analysé la probabilité de tirer au hasard un changement de variables qui met un idéal en position de Noether [Giu+00].

La mise en position de Noether est un ingrédient majeur de l'algorithme de résolution géométrique, présenté dans le chapitre suivant, et qui permet de résoudre des systèmes d'équations polynomiales sans calculer de bases standard. Les techniques de preuve de la Section 27.3, sur les paramétrisations des variétés projectives, sont issues de l'article de Durvye et Lecerf [DL07].

Bibliographie

DL07 Durvye, Clémence et Grégoire Lecerf (2007). « A concise proof of the Kronecker polynomial system solver from scratch ». In : *Expositiones Mathematicae*, vol. 26, n°2, p. 101–139.

GH93 Giusti, Marc et Joos Heintz (1993). « La détermination des points isolés et de la dimension d'une variété algébrique peut se faire en temps polynomial ». In : *Computational algebraic geometry and commutative algebra (Cortona, 1991)*. Vol. XXXIV. Sympos. Math. Cambridge : Cambridge Univ. Press, p. 216–256.

Giu+00 Giusti, Marc, Klemens Hägele, Grégoire Lecerf, Joël Marchand et Bruno Salvy (2000). « The Projective Noether Maple package : computing the dimension of a projective variety ». In : *Journal of Symbolic Computation*, vol. 30, n°3, p. 291–307.

Giu88 Giusti, Marc (1988). « Combinatorial dimension theory of algebraic varieties ». In : *Journal of Symbolic Computation*, vol. 6, n°2-3. Special issue on Computational aspects of commutative algebra, p. 249–265.

28. Résolution géométrique

Résumé

Dans ce chapitre nous présentons une version simplifiée de l'algorithme de résolution géométrique, qui permet de résoudre efficacement une suite régulière réduite de polynômes homogènes à coefficients dans un corps de caractéristique zéro ou suffisamment grande. L'algorithme s'exprime simplement ; il repose sur les paramétrisations sous forme de Kronecker d'idéaux de dimensions quelconques présentées au chapitre précédent ; il est indépendant de tout calcul de base standard et peut tirer parti d'un programme évaluant efficacement le système donné en entrée.

28.1 Introduction

Tout au long de ce chapitre le corps de base, noté k, est supposé être de caractéristique zéro ou suffisamment grande. Nous n'entrerons pas dans ces détails pour alléger la présentation. Le système à résoudre est noté $f_1 = \cdots = f_n = 0$, où f_i est un polynôme homogène de $\mathrm{R} = k[\mathrm{X}_0, \ldots, \mathrm{X}_n]$ de degré total d_i. On suppose que $f_1, \ldots, f_n$ forment une *suite régulière* (Définition 26.10), et que les *idéaux intermédiaires* $\mathrm{I}_i = (f_1, \ldots, f_i)$ sont radicaux, pour tout i de 1 à n. On dit que $f_1, \ldots, f_n$ forment une suite *régulière réduite*.

Sous ces hypothèses, nous décrivons ici l'algorithme dit de *résolution géométrique*, qui résout un tel système de façon indépendante des méthodes par bases standard vues dans les chapitres précédents. En général, sur des problèmes concrets, l'algorithme de résolution géométrique s'avère être asymptotiquement plus rapide que les méthodes par bases standard ; la discussion est délicate à ce stade et reportée en fin de chapitre.

Une fois les positions de Noether acquises, à chaque idéal intermédiaire I_i est associée son extension I'_i dans l'anneau $\mathrm{K}_{n-i}[\mathrm{X}_{n-i+1}, \ldots, \mathrm{X}_n]$, avec $\mathrm{K}_{n-i} = k(\mathrm{X}_0, \ldots, \mathrm{X}_{n-i})$. Cette construction a été abordée en Section 27.3. Le Théorème 27.16 assure que I'_i

est un idéal radical non homogène de dimension zéro (dans le cadre affine de la Définition 25.2) et qu'une paramétrisation par un élément primitif linéaire homogène est possible avec de bonnes propriétés. En particulier, la fin de Section 27.4 explique en quoi une telle paramétrisation constitue une représentation de $\mathbf{V}_{\bar{k}}(\mathrm{I}_i)$. Cette représentation est utilisée par l'algorithme pour représenter chaque I'_i. La résolution est effectuée de *manière incrémentale*, c'est-à-dire en ajoutant les équations une par une : l'algorithme calcule ainsi une paramétrisation pour I'_1, puis I'_2, I'_3, etc, jusqu'à I'_n.

Le reste de cette introduction est consacrée aux aspects probabilistes liés à l'algorithme, à quelques exemples montrant les objets manipulés en cours d'exécution, et aux structures de données utilisées pour représenter les polynômes en entrée. Dans la prochaine section nous décrivons les principales formules qui sont au cœur de l'approche incrémentale de résolution. Ensuite nous verrons comment rendre cette approche efficace : le principe méthodologique utilisé s'apparente aux méthodes par évaluation et interpolation du Chapitre 5, mais où les phases d'interpolation sont confiées à un opérateur de Newton.

Aspects probabilistes

Sous les hypothèses décrites, la dimension de l'idéal intermédiaire I_i est $n-i$, par le Corollaire 26.8. L'algorithme de résolution géométrique commence par choisir un changement linéaire de coordonnées qui met les idéaux intermédiaires I_i en position de Noether simultanée (Définition 27.2). En pratique on peut choisir un tel changement de variables au hasard avec une très forte probabilité de succès grâce au Corollaire 27.7. Pour cette raison, l'algorithme de résolution géométrique est probabiliste. D'autres aspects probabilistes comme les choix d'éléments primitifs interviennent aussi.

Il est néanmoins légitime de se demander dès maintenant s'il est possible de garantir qu'un résultat calculé par l'algorithme de résolution est correct ou non. Une condition suffisante facile à tester fait l'objet du résultat suivant, que nous admettrons.

Lemme 28.1 Soient $f_1,\ldots,f_n$ des polynômes homogènes de R de degrés respectifs $d_1,\ldots,d_n$. Considérons donnés une forme linéaire $u = \lambda_1\mathrm{X}_1 + \cdots + \lambda_n\mathrm{X}_n$ et des polynômes $q, v_1,\ldots,v_n$ de $k(\mathrm{X}_0)[\mathrm{T}]$ tels que :

- q est sans carré de degré $d_1\cdots d_n$;
- pour tout i de 1 à n, l'évaluation de $f_i(\mathrm{X}_0, v_1(\mathrm{T}),\ldots,v_n(\mathrm{T}))$ se réduit à 0 modulo $q(\mathrm{T})$;
- $\mathrm{J}(\mathrm{X}_0, v_1(\mathrm{T}),\ldots,v_n(\mathrm{T}))$ est inversible modulo $q(\mathrm{T})$, où $\mathrm{J}(\mathrm{X}_0,\mathrm{X}_1,\ldots,\mathrm{X}_n)$ représente le déterminant de la matrice jacobienne $\left(\frac{\partial f_i}{\partial \mathrm{X}_j}\right)_{\substack{1\le i\le n\\ 1\le j\le n}}$.

Alors, $\mathrm{I} = (f_1,\ldots,f_n)$ est en position de Noether, radical, de dimension 0, et les polynômes $q, v_1,\ldots,v_n$ forment la paramétrisation de I par l'élément primitif u.

Le résultat calculé peut donc toujours être vérifié. Néanmoins, lorsque la vérification échoue, cela peut signifier que des mauvais choix ont été faits, ou bien que le système n'admet pas $d_1\cdots d_n$ solutions distinctes. Ce second cas peut être conforté en multipliant le nombre d'exécutions de l'algorithme pour différents choix aléatoires. Nous

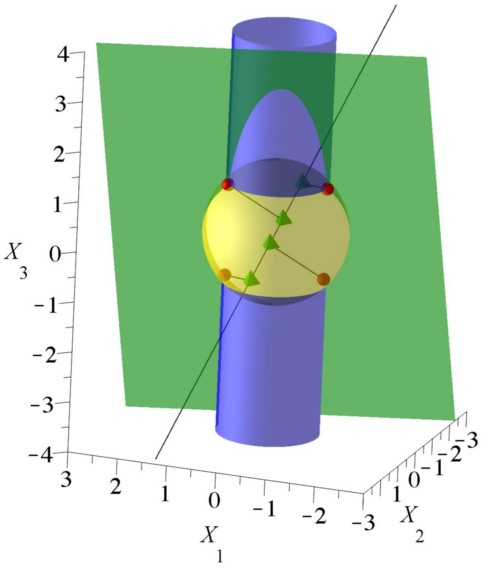


FIGURE 28.1 – Les quatre points d'intersection d'une sphère, d'un cylindre, et d'un plan, matérialisés par des petites sphères. Leurs projections orthogonales sur la droite portant les valeurs de l'élément primitif sont représentées par des petits tétraèdres.

abordons très brièvement en fin de chapitre la question de certifier que $f_1, \ldots, f_n$ n'est pas une suite régulière réduite.

Exemple

Jusqu'à la fin de cette section, nous utilisons l'exemple suivant, où $n = 3$.

Exemple 28.1 On considère les polynômes

$$f_1 = X_1^2 + X_2^2 + X_3^2 - 2X_0^2, \quad f_2 = X_1^2 + X_2^2 - X_0^2, \quad f_3 = X_1 - 3X_2 + X_3.$$

Les solutions du système $f_1 = f_2 = f_3 = 0$, précisément $\mathbf{V}_{\bar{k}}(I_3)$, sont particulièrement simples dans cet exemple et peuvent être explicitées :

$$\mathbf{V}_{\bar{k}}(I_3) = \left\{(1:1:0:-1), (1:-1:0:1), (1:\frac{4}{5}:\frac{3}{5}:1), (1:-\frac{4}{5}:-\frac{3}{5}:-1)\right\}.$$

Cette variété n'a pas de point à l'infini, et sa partie affine est la variété affine $\mathbf{V}_{\bar{k}}(I_3 + (X_0 - 1))$, présentée Figure 28.1.

L'algorithme de résolution présente $\mathbf{V}_{\bar{k}}(I_3)$ sous la forme d'une paramétrisation

par élément primitif sous forme de Kronecker (Définition 25.5) :

$$u = \mathrm{X}_1 + 2\mathrm{X}_2 - 3\mathrm{X}_3, \quad q(\mathrm{T}) = \mathrm{T}^4 - 17\mathrm{T}^2\mathrm{X}_0^2 + 16\mathrm{X}_0^4 = 0,$$
$$\mathrm{X}_1 = \frac{8}{5}\mathrm{X}_0^2\frac{4\mathrm{T}^2 + 11\mathrm{X}_0^2}{q'(\mathrm{T})}, \quad \mathrm{X}_2 = -\frac{6}{5}\mathrm{X}_0^2\frac{\mathrm{T}^2 - 16\mathrm{X}_0^2}{q'(\mathrm{T})}, \quad \mathrm{X}_3 = -10\mathrm{X}_0^2\frac{\mathrm{T}^2 - 4\mathrm{X}_0^2}{q'(\mathrm{T})}.$$

Ainsi, en substituant X_0 par 1 dans cette paramétrisation, le polynôme $q(\mathrm{T})$ admet quatre racines complexes, et pour chacune d'entre elles, les coordonnées $\mathrm{X}_1, \mathrm{X}_2, \mathrm{X}_3$ ne peuvent prendre qu'une valeur, donnée par les trois dernières équations de la paramétrisation. Le Corollaire 27.18 assure que les solutions du système sont toutes obtenues de cette manière.

Le polynôme $q(\mathrm{T})$ est ici le polynôme minimal de l'élément primitif u, c'est-à-dire que ses racines sont les valeurs que prend l'élément primitif en chacun des points de $\mathbf{V}_{\bar{k}}(\mathrm{I}_3)$. Cette propriété est illustrée dans la Figure 28.1, qui montre la partie affine d'une droite orthogonale au plan d'équation $\mathrm{X}_1 + 2\mathrm{X}_2 - 3\mathrm{X}_3 = 0$, ainsi que les projections orthogonales, distinctes, des points de $\mathbf{V}_{\bar{k}}(\mathrm{I}_3)$ sur cette droite.

Pour parvenir à la paramétrisation de I_3' ci-dessus, l'algorithme repose sur les paramétrisations par élément primitif de l'idéal I_1' engendré par f_1 dans $k(\mathrm{X}_0, \mathrm{X}_1, \mathrm{X}_2)[\mathrm{X}_3]$, puis de l'idéal I_2' engendré par f_1, f_2 dans $k(\mathrm{X}_0, \mathrm{X}_1)[\mathrm{X}_2, \mathrm{X}_3]$ et enfin de I_3' l'extension de I_3 dans $k(\mathrm{X}_0)[\mathrm{X}_1, \mathrm{X}_2, \mathrm{X}_3]$.

Exemple 28.2 Explicitement, ces paramétrisations intermédiaires sont

$$\mathrm{I}_1' : \quad u = -3\mathrm{X}_3, \qquad \begin{cases} q(\mathrm{T}) = \mathrm{T}^2 - 9(2\mathrm{X}_0^2 + \mathrm{X}_1^2 + \mathrm{X}_2^2), \\ q'(\mathrm{T})\mathrm{X}_3 = 6(-2\mathrm{X}_0^2 + \mathrm{X}_1^2 + \mathrm{X}_2^2), \end{cases}$$

$$\mathrm{I}_2' : \quad u = 2\mathrm{X}_2 - 3\mathrm{X}_3, \qquad \begin{cases} q(\mathrm{T}) = \mathrm{T}^4 - (26\mathrm{X}_0^2 - 8\mathrm{X}_1^2)\mathrm{T}^2 + (5\mathrm{X}_0^2 + 4\mathrm{X}_1^2)^2, \\ q'(\mathrm{T})\mathrm{X}_2 = 8(\mathrm{X}_0^2 - \mathrm{X}_1^2)(\mathrm{T}^2 + 5\mathrm{X}_0^2 + 4\mathrm{X}_1^2), \\ q'(\mathrm{T})\mathrm{X}_3 = -12\mathrm{X}_0^2(\mathrm{T}^2 - 5\mathrm{X}_0^2 - 4\mathrm{X}_1^2). \end{cases}$$

Une première approche, calculant effectivement ces paramétrisations, est présentée en Section 28.2. Cependant, les paramétrisations faisant intervenir les variables libres deviennent rapidement grosses, et l'algorithme de résolution évite de les calculer. L'idée part de la Proposition 27.17, qui assure que pour presque toutes les valeurs a_0, a_1 et a_2 prises respectivement par les variables X_0, X_1 et X_2, les spécialisations de ces paramétrisations donnent des paramétrisations des variétés affines de dimension 0 des idéaux $\mathrm{I}_1 + (\mathrm{X}_0 - a_0, \mathrm{X}_1 - a_1, \mathrm{X}_2 - a_2)$ puis $\mathrm{I}_2 + (\mathrm{X}_0 - a_0, \mathrm{X}_1 - a_1)$. L'algorithme principal, décrit en Section 28.4, prend donc en entrée un tel point (a_0, a_1, a_2) et calcule juste ces spécialisations.

Exemple 28.3 Avec le choix $(a_0, a_1, a_2) = (1, 1/2, 0)$, les calculs intermédiaires de

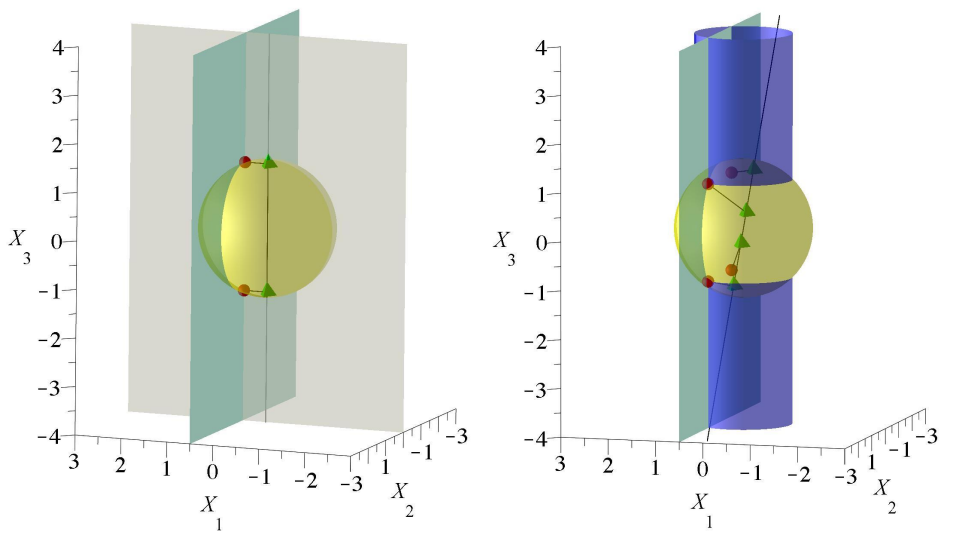


FIGURE 28.2 – Les paramétrisations de $(f_1, X_0 - 1, X_1 - 1/2, X_2)$ et $(f_1, f_2, X_0 - 1, X_1 - 1/2)$.

l'algorithme produisent donc simplement

$$I_1' : \ u = -3X_3, \begin{cases} q(T) = T^2 - \frac{63}{4}, \\ q'(T)X_3 = -\frac{21}{2}, \end{cases} \qquad I_2' : \ u = 2X_2 - 3X_3, \begin{cases} q(T) = T^4 - 24T^2 + 36, \\ q'(T)X_2 = 6T^2 + 36, \\ q'(T)X_3 = -12T^2 - 72. \end{cases}$$

$$I_3' : \ u = X_1 + 2X_2 - 3X_3, \begin{cases} q(T) = T^4 - 17T^2 + 16 = 0, \\ q'(T)X_1 = \frac{8}{5}(4T^2 + 11), \\ q'(T)X_2 = -\frac{6}{5}(T^2 - 16), \\ q'(T)X_3 = -10(T^2 - 4). \end{cases}$$

Les variétés calculées lors de ces étapes intermédiaires sont données en Figure 28.2.

L'étape incrémentale permettant le passage d'une telle paramétrisation à la suivante nécessite de « dé-spécialiser » la dernière variable libre, ce qui est réalisé par une itération de Newton, décrite en Section 28.3.

Exemple 28.4 À partir de la spécialisation de la paramétrisation de I_1', l'algorithme de remontée obtient une paramétrisation de $I_1 + (X_1 - X_0/2)$, que l'on peut montrer être un idéal de dimension 1 : la partie affine de la variété correspondante est un cercle tracé sur la sphère dans la partie gauche de la Figure 28.2. Les points de $I_2 + (X_0 - 1, X_1 - 1/2)$ sont alors l'intersection de ce cercle avec le cylindre défini par la deuxième équation $f_2 = 0$. Pour continuer la résolution, l'algorithme remonte la variable X_1 pour obtenir une paramétrisation de I_2, dont la partie affine correspond aux deux cercles de l'intersection de la sphère et du cylindre dans la

partie droite de la Figure 28.2. Finalement les solutions du système $f_1 = f_2 = f_3 = 0$ sont l'intersection de ces deux cercles avec le plan d'équation $f_3 = 0$.

Structures de données pour les polynômes en entrée

Jusqu'ici dans les algorithmes de cette partie, les polynômes étaient donnés dans leur *représentation dense*, qui consiste à stocker le degré total et la liste de tous les coefficients (même nuls) des monômes de degrés plus petits. Par exemple, pour un polynôme homogène de degré d en n variables, on doit stocker $\binom{d+n}{n}$ coefficients. L'inconvénient de cette représentation est qu'elle oublie toute structure du calcul qui a mené à la construction du polynôme.

Exemple 28.5 En mécanique il est fréquent que des normes de vecteurs interviennent dans les équations. Si (p_1, p_2, p_3) est un vecteur dont les coordonnées sont des expressions polynomiales p_i, il est préférable de ne pas développer le carré de la norme $p_1^2 + p_2^2 + p_3^2$, afin de pouvoir évaluer numériquement cette norme pour une spécialisation donnée des variables.

Dans l'étude des équilibres chimiques, les équations provenant des lois d'action de masse possèdent très peu de monômes non nuls, et une représentation dense n'est pas bien adaptée pour évaluer ces équations.

Les méthodes de résolution numérique des systèmes polynomiaux, par subdivision, ou à l'aide de l'opérateur de Newton, passent l'essentiel de leur temps de calcul à évaluer les polynômes en entrée sur des valeurs numériques. Plus les polynômes s'évaluent bien, plus la résolution est rapide. La représentation dense n'est alors pas appropriée. D'une manière symbolique analogue, l'algorithme de résolution géométrique décrit dans ce chapitre passe aussi son temps à évaluer les équations en entrée en des points appartenant à des algèbres.

Afin de conserver leurs propriétés naturelles en terme d'évaluation, les polynômes $f_1, \ldots, f_n$ en entrée de l'algorithme de résolution sont donnés sous forme de DAG, pour *Directed Acyclic Graph* en anglais et graphe orienté acyclique en français (Définition 12.1). Les nœuds du graphe sont binaires et représentent les opérations arithmétiques $(+, -, \times)$. Les feuilles (les nœuds sans arête sortante) sont des éléments de k ou des variables X_i. Les nœuds sans parent sont les sorties du calcul représentant les polynômes $f_1, \ldots, f_n$. La taille du graphe (nombre de nœuds) définit la *complexité d'évaluation* du calcul. Il s'agit du nombre d'opérations arithmétiques nécessaires pour évaluer les polynômes en entrée sur un point.

Notons que cette structure de donnée présente l'avantage de partager les sous-expressions communes, et permet ainsi d'éviter d'évaluer plusieurs fois une même expression sur des valeurs identiques.

Exemple 28.6 Tout polynôme peut être représenté par un DAG. Par exemple, le schéma de Horner consiste à calculer $P = \sum_{i=0}^{d} a_i X^i$ par $P = ((a_d X + a_{d-1})X + a_{d-2})\cdots)X + a_0$. Il code tout polynôme à une variable de degré d par un DAG de complexité d'évaluation en $O(d)$. À l'inverse, le polynôme $(X + 1)^{2^d}$ se code

beaucoup mieux par le DAG $(\cdots(((X+1)^2)^2)^2\cdots)^2$ que par sa représentation sur la base monomiale.

Définition 28.1 La complexité d'évaluation d'un polynôme est le minimum des complexités d'évaluation des DAG qui le représentent.

Lemme 28.2 Un polynôme homogène de R de degré d a une complexité d'évaluation $O(\binom{d+n}{n})$.

Démonstration. La borne correspond à l'évaluation du polynôme en utilisant naïvement sa représentation dense qui est justement de taille $\binom{n+d}{n}$. ■

Exemple 28.7 Soit $M = (X_{i,j})$ la matrice générique carré de taille n (c'est-à-dire que ses coefficients sont des variables distinctes). Alors le polynôme déterminant de M se code mal par sa représentation sur la base monomiale car il a $n!$ monômes. Par contre, il se code en complexité d'évaluation $O(n^4)$ par l'algorithme de Berkowitz (Théorème 8.9). Cette exemple est important car il illustre que les polynômes d'élimination, dont le déterminant est un bon représentant, s'évaluent bien.

La complexité de l'algorithme de résolution géométrique s'exprime en terme du produit des degrés d_i des polynômes f_i, du nombre de variables homogènes n, et de la complexité d'évaluation, notée L, du DAG utilisé pour représenter $f_1,\ldots,f_n$.

Soulignons que pour obtenir une position de Noether, il suffit d'appliquer un changement de variables linéaire aux polynômes en entrée, ce qui augmente la complexité d'évaluation L de $O(n^2)$, et reste négligeable sur le coût total de l'algorithme.

28.2 Approche incrémentale

L'algorithme développé dans ce chapitre procède de manière incrémentale, en ajoutant une nouvelle équation à chaque étape. Informellement, on résout d'abord l'équation $f_1 = 0$, puis le système $f_1 = f_2 = 0$, puis $f_1 = f_2 = f_3 = 0$, etc. Les ensembles de solutions sont représentés par éléments primitifs. Dans l'Exemple 28.2, l'étape incrémentale décrite dans cette section permet ainsi le calcul de la paramétrisation de I'_2 à partir de celle de I'_1.

Hypothèses et notations

Jusqu'à la fin du chapitre nous travaillons sous les hypothèses suivantes :

(H_1) la suite $f_1,\ldots,f_n$ est régulière ;
(H_2) la suite $f_1,\ldots,f_n$ est *réduite*, c'est-à-dire que chacun des idéaux I_i est radical ;
(H_3) la suite $f_1,\ldots,f_n$ est en *position de Noether simultanée* (Définition 27.2).

Nous utilisons les notations suivantes, pour $i \in \{0,\dots,n\}$:

$$\begin{aligned} r &= n-i, & \mathrm{I}_i &= (f_1,\dots,f_i),\\ \mathrm{A}_r &= k[\mathrm{X}_0,\dots,\mathrm{X}_r], & \mathrm{K}_r &= k(\mathrm{X}_0,\dots,\mathrm{X}_r),\\ \mathrm{B}_r &= \mathrm{R}/\mathrm{I}_i, & \mathrm{E}_r &= \mathrm{K}_r[\mathrm{X}_{r+1},\dots,\mathrm{X}_n]/\mathrm{I}_i' \end{aligned}$$

où I_i' représente l'extension de I_i à $\mathrm{K}_r[\mathrm{X}_{r+1},\dots,\mathrm{X}_n]$.

Entrée et sortie de l'étape

À l'étape i de la résolution, la paramétrisation de I_i' sous forme de Kronecker est constituée de :

— l'élément primitif $u = \lambda_{r+1}\mathrm{X}_{r+1} + \cdots + \lambda_n\mathrm{X}_n$;
— le polynôme minimal $q(\mathrm{T}) \in \mathrm{A}_r[\mathrm{T}]$ de la multiplication par u dans E_r ;
— les polynômes $w_{r+1},\dots,w_n$ de $\mathrm{A}_r[\mathrm{T}]$ tels que

$$\mathrm{I}_i' = (q(u), q'(u)\mathrm{X}_{r+1} - w_{r+1}(u),\dots,q'(u)\mathrm{X}_n - w_n(u)).$$

On note $v_j(\mathrm{T}) = q'(\mathrm{T})^{-1}w_j(\mathrm{T}) \bmod q(\mathrm{T})$. En d'autres termes, l'application

$$\begin{aligned} &\mathrm{E}_r \to \mathrm{K}_r[\mathrm{T}]/(q(\mathrm{T}))\\ &\mathrm{X}_j \mapsto v_j(\mathrm{T}) \quad \text{pour } r+1 \le j \le n \end{aligned} \tag{28.1}$$

est un isomorphisme de K_r-algèbres. En particulier le degré de q est $\delta_i = \dim_{\mathrm{K}_r}\mathrm{E}_r$.

Comme $f_1,\dots,f_n$ forment une suite régulière, le Théorème 26.7 donne $\deg \mathrm{I}_i = d_1\cdots d_i$. Par ailleurs, le Théorème 26.5 assure que ce degré est également la dimension de E_r :

$$\delta_i = \deg \mathrm{I}_i. \tag{28.2}$$

On considère ensuite l'équation suivante $f_{i+1} = 0$. Le but de cette section est de décrire un algorithme qui calcule la paramétrisation par l'élément primitif $\mathrm{U} = \lambda_r\mathrm{X}_r + \cdots + \lambda_n\mathrm{X}_n$ de I_{i+1}'. Les polynômes de la nouvelle paramétrisation sous forme de Kronecker sont notés $\mathrm{Q}, \mathrm{W}_r,\dots,\mathrm{W}_n$. Pour alléger les notations, l'indice i n'apparaît pas dans les paramétrisations : la paramétrisation de l'étape i est notée en minuscules, et celle de l'étape $i+1$ en majuscules. (Le nombre de polynômes de la paramétrisation indique bien sûr l'idéal intermédiaire I_i concerné).

Élément primitif générique

Le calcul de la nouvelle paramétrisation est réalisé avec la technique de l'élément primitif « générique » de la Section 25.2. Nous introduisons donc les nouvelles variables $\Lambda_1,\dots,\Lambda_n$, et les notations suivantes :

$$\begin{aligned} k_\Lambda &= k(\Lambda_1,\dots,\Lambda_n), & \mathrm{R}_\Lambda &= k_\Lambda[\mathrm{X}_0,\dots,\mathrm{X}_n],\\ \mathrm{A}_{\Lambda,r} &= k_\Lambda[\mathrm{X}_0,\dots,\mathrm{X}_r], & \mathrm{K}_{\Lambda,r} &= k_\Lambda(\mathrm{X}_0,\dots,\mathrm{X}_r),\\ \mathrm{B}_{\Lambda,r} &= \mathrm{R}_\Lambda/\mathrm{I}_{\Lambda,i}, & \mathrm{E}_{\Lambda,r} &= \mathrm{K}_{\Lambda,r}[\mathrm{X}_{r+1},\dots,\mathrm{X}_n]/\mathrm{I}_{\Lambda,i}', \end{aligned}$$

où $\mathrm{I}_{\Lambda,i}$ (resp. $\mathrm{I}_{\Lambda,i}'$) représente l'extension de I_i dans R_Λ (resp. dans $\mathrm{K}_{\Lambda,r}[\mathrm{X}_{r+1},\dots,\mathrm{X}_n]$). Aussi, on note I_i^Λ l'extension de I_i dans $\mathrm{R}^\Lambda = k[\Lambda_1,\dots,\Lambda_n,\mathrm{X}_0,\dots,\mathrm{X}_n]$.

Cette extension avec les Λ_i ne change pas grand chose à la situation, grâce au Lemme 27.15 qui montre les propriétés suivantes :

— la suite $f_1,\dots,f_n$ est régulière dans R_Λ ;
— $\mathrm{I}_{\Lambda,i}$ est radical pour tout $i\in\{1,\dots,n\}$;
— pour tout $i\in\{1,\dots,n\}$, on a $\dim \mathrm{I}_{\Lambda,i}=\dim \mathrm{I}_i=r$, et $\deg \mathrm{I}_{\Lambda,i}=\deg \mathrm{I}_i=\delta_i=\dim_{\mathrm{K}_{\Lambda,r}}\mathrm{E}_r$.

La paramétrisation de I'_i est donc aussi une paramétrisation de $\mathrm{I}'_{\Lambda,i}$:

$$\mathrm{I}'_{\Lambda,i}=(q(u),q'(u)\mathrm{X}_{r+1}-w_{r+1}(u),\dots,q'(u)\mathrm{X}_n-w_n(u)).$$

De façon équivalente, l'application

$$\begin{aligned}\mathrm{E}_{\Lambda,r}&\to \mathrm{K}_{\Lambda,r}[\mathrm{T}]/(q(\mathrm{T}))\\ \mathrm{X}_j&\mapsto v_j(\mathrm{T})\quad \text{pour } r+1\le j\le n\end{aligned}\tag{28.3}$$

est un isomorphisme de $\mathrm{K}_{\Lambda,r}$-algèbres.

Remontée de l'élément primitif générique et de la paramétrisation

Pour calculer la paramétrisation « générique » de $\mathrm{I}'_{\Lambda,i}$, c'est-à-dire par l'élément primitif $u_\Lambda=\Lambda_{r+1}\mathrm{X}_{r+1}+\cdots+\Lambda_n\mathrm{X}_n$, une première solution consiste à calculer le polynôme caractéristique q_Λ de u_Λ dans $\mathrm{K}_{\Lambda,r}[\mathrm{T}]/(q(\mathrm{T}))$ par la formule

$$q_\Lambda(\mathrm{T})=\mathrm{Res}_\mathrm{S}\left(q(\mathrm{S}),\mathrm{T}-u_\Lambda(v_{r+1}(\mathrm{S}),\dots,v_n(\mathrm{S}))\right).$$

De là on déduit les paramétrisations par $w_{\Lambda,j}(\mathrm{T})=-\partial q_\Lambda(\mathrm{T})/\partial\Lambda_j$; le Lemme 27.15 donne alors :

$$\mathrm{I}'_{\Lambda,i}=(q_\Lambda(u_\Lambda),q'_\Lambda(u_\Lambda)\mathrm{X}_{r+1}-w_{\Lambda,r+1}(u_\Lambda),\dots,q'_\Lambda(u_\Lambda)\mathrm{X}_n-w_{\Lambda,n}(u_\Lambda)),$$

et

$$\mathrm{I}_i^\Lambda\supseteq(q_\Lambda(u_\Lambda),q'_\Lambda(u_\Lambda)\mathrm{X}_{r+1}-w_{\Lambda,r+1}(u_\Lambda),\dots,q'_\Lambda(u_\Lambda)\mathrm{X}_n-w_{\Lambda,n}(u_\Lambda)).$$

Cependant, plutôt que de calculer avec des fractions rationnelles en les Λ_j, il est possible de se contenter de développements en séries au premier ordre au voisinage de $(\lambda_1,\dots,\lambda_n)$. Il est commode d'introduire l'idéal maximal

$$\mathfrak{L}=(\Lambda_1-\lambda_1,\dots,\Lambda_n-\lambda_n).$$

Pour calculer q_Λ modulo $\mathfrak{L}^2$ il suffit d'utiliser le Corollaire 25.16 : au signe près, les paramétrisations w_i sont les coefficients des termes linéaires en $(\Lambda_i-\lambda_i)$, fournissant sans calcul

$$q_\Lambda(\mathrm{T})=q(\mathrm{T})-(\Lambda_{r+1}-\lambda_{r+1})w_{r+1}(\mathrm{T})-\cdots-(\Lambda_n-\lambda_n)w_n(\mathrm{T})\bmod\mathfrak{L}^2.$$

Il faut également calculer les paramétrisations $v_{\Lambda,j}=w_{\Lambda,j}/q'_\Lambda$. Pour cela, on peut interpréter le calcul précédent en terme de remontée de Hensel : en posant

$$\Delta(\mathrm{T})=q'(\mathrm{T})^{-1}(q_\Lambda(\mathrm{T})-q(\mathrm{T}))\bmod q(\mathrm{T})\bmod\mathfrak{L}^2$$

on a $q(\mathrm{T}+\Delta(\mathrm{T}))=0\bmod q_\Lambda(\mathrm{T})\bmod\mathfrak{L}^2$. Par ailleurs, de $q'(u)\mathrm{X}_j-w_j(u)\in\mathrm{I}_i$ on déduit que $\mathrm{X}_j-v_j(u_\Lambda+\Delta(u_\Lambda))$ appartient à l'extension de I_i à

$$(k[\Lambda_1,\dots,\Lambda_n]/\mathfrak{L}^2)(\mathrm{X}_0,\dots,\mathrm{X}_r)[\mathrm{X}_{r+1},\dots,\mathrm{X}_n].$$

On obtient ainsi la formule

$$v_{\Lambda,j}(\mathrm{T}) = v_j(\mathrm{T}) + (v'_j(\mathrm{T})\Delta(\mathrm{T}) \bmod q(\mathrm{T})) \bmod \mathfrak{L}^2,$$

puis

$$w_{\Lambda,j}(\mathrm{T}) = q'_\Lambda(\mathrm{T})v_{\Lambda,j}(\mathrm{T}) \bmod q_\Lambda(\mathrm{T}) \bmod \mathfrak{L}^2.$$

Calcul de la paramétrisation suivante

Pour expliquer comment la paramétrisation suivante est calculée, commençons par montrer que le calcul du polynôme minimal P de la multiplication par X_r dans E_{r-1}, lorsque X_r est primitif, peut s'obtenir par la formule

$$\mathrm{P}(\mathrm{X}_r) = \mathrm{Res}_\mathrm{T}\,(q(\mathrm{T}), f_{i+1}(\mathrm{X}_0,\ldots,\mathrm{X}_r, v_{r+1}(\mathrm{T}),\ldots,v_n(\mathrm{T}))).$$

En effet, l'isomorphisme (28.1) et le Lemme 6.9 assurent que P est le coefficient constant du polynôme caractéristique de la multiplication par f_{i+1} dans E_r. En particulier, par la Proposition 27.9, ce polynôme P est homogène de degré total $d_{i+1}\delta_i = \delta_{i+1}$. Le Lemme 27.11 donne ensuite que P est unitaire en X_r. Et enfin le Corollaire 27.14 assure que P appartient à I_{i+1}. Par conséquent, si X_r est primitif pour I'_{i+1} alors P est bien le polynôme minimal de la multiplication par X_r dans E_{r-1}.

Exemple 28.8 Dans l'Exemple 28.2, en partant de la paramétrisation de I'_2, ce calcul de P commence par la réduction modulo $q(\mathrm{T})$ de f_3 évalué en

$$\left(\mathrm{X}_0, \mathrm{X}_1, \frac{8(\mathrm{X}_0^2-\mathrm{X}_1^2)(\mathrm{T}^2-2\mathrm{X}_1\mathrm{T}+5\mathrm{X}_0^2+5\mathrm{X}_1^2)}{q'(\mathrm{T})}, \frac{-12\mathrm{X}_0^2(\mathrm{T}^2-2\mathrm{X}_1\mathrm{T}-5\mathrm{X}_0^2-4\mathrm{X}_1^2)}{q'(\mathrm{T})}\right),$$

ce qui donne

$$\frac{-7\mathrm{T}^3+(127\mathrm{X}_0^2-100\mathrm{X}_1^2)\mathrm{T}^2+12\mathrm{X}_1(5\mathrm{X}_0^2+4\mathrm{X}_1^2)}{12(5\mathrm{X}_0^2+4\mathrm{X}_1^2)}.$$

Malgré le dénominateur, le raisonnement ci-dessus affirme que le résultant avec $q(\mathrm{T})$ est un polynôme, et en effet, le calcul fournit

$$\mathrm{P}(\mathrm{X}_1) = 100\mathrm{X}_1^4 - 164\mathrm{X}_1^2\mathrm{X}_0^2 + 64\mathrm{X}_0^2,$$

qui est sans carré, de degré $4 = 4\times1$, et redonne bien le terme constant du polynôme minimal de f_3 trouvé à l'Exemple 27.4.

Pour obtenir une paramétrisation de E_{r-1}, nous partons de cette idée, mais en « déformant » la variable X_r de sorte à obtenir le polynôme minimal de $\mathrm{U}_\Lambda = \Lambda_r\mathrm{X}_r + u_\Lambda$ dans E_{r-1} à la place de P.

Proposition 28.3 Sous les hypothèses (H_1)–(H_3), soient

$$\begin{aligned}
\tilde{q}_\Lambda(T) &= q_\Lambda(T)|_{X_r \mapsto (Y-T)/\Lambda_r},\\
\tilde{p}_\Lambda(T) &= (q'_\Lambda(T)\big|_{X_r \mapsto (Y-T)/\Lambda_r})^{-1} \bmod \tilde{q}_\Lambda(T),\\
\tilde{w}_{\Lambda,j}(T) &= w_{\Lambda,j}(T)\big|_{X_r \mapsto (Y-T)/\Lambda_r}, \text{ pour } r+1 \le j \le n,\\
\tilde{v}_{\Lambda,j}(T) &= \tilde{p}_\Lambda(T)\tilde{w}_{\Lambda,j}(T) \bmod \tilde{q}_\Lambda(T), \text{ pour } r+1 \le j \le n,\\
\tilde{h}(T) &= f_{i+1}(X_0,\dots,X_{r-1},(Y-T)/\Lambda_r,\tilde{v}_{\Lambda,r+1}(T),\dots,\tilde{v}_{\Lambda,n}(T)) \bmod \tilde{q}_\Lambda(T),
\end{aligned}$$

alors le résultant $Q_\Lambda(Y) = \mathrm{Res}_T(\tilde{q}_\Lambda(T),\tilde{h}(T))$ est le polynôme minimal de la multiplication par $U_\Lambda = \Lambda_r X_r + u_\Lambda$ dans $E_{\Lambda,r-1}$. Il est de degré total δ_{i+1}, unitaire en Y, sans carré, et satisfait $Q_\Lambda(U_\Lambda) \in I_{i+1}^\Lambda$.

Démonstration. Nous utilisons des « ˜ » pour noter les objets obtenus après le changement de variable $X_r \mapsto (Y-u_\Lambda)/\Lambda_r$. On note donc $\tilde{f}_{\Lambda,i} = f_i|_{X_r \mapsto (Y-u_\Lambda)/\Lambda_r}$, et $\tilde{I}_{\Lambda,i} = (\tilde{f}_{\Lambda,1},\dots,\tilde{f}_{\Lambda,i})$, et on décline les notations en conséquence :

$$\begin{aligned}
\tilde{R}_\Lambda &= k_\Lambda[X_0,\dots,X_{r-1},Y,X_{r+1},\dots,X_n],\\
\tilde{A}_{\Lambda,r} &= k_\Lambda[X_0,\dots,X_{r-1},Y], & \tilde{K}_{\Lambda,r} &= k_\Lambda(X_0,\dots,X_{r-1},Y),\\
\tilde{B}_{\Lambda,r} &= \tilde{R}_\Lambda/\tilde{I}_{\Lambda,i}, & \tilde{E}_{\Lambda,r} &= \tilde{K}_{\Lambda,r}[X_{r+1},\dots,X_n]/\tilde{I}'_{\Lambda,i},
\end{aligned}$$

où $\tilde{I}'_{\Lambda,i}$ est l'extension de $\tilde{I}_{\Lambda,i}$ dans $\tilde{K}_{\Lambda,r}[X_{r+1},\dots,X_n]$.

La suite $\tilde{f}_{\Lambda,1},\dots,\tilde{f}_{\Lambda,n}$ reste régulière dans $\tilde{R}_\Lambda$ et les idéaux intermédiaires sont radicaux. Par ailleurs, appliquer le changement de variable inversible $X_r \mapsto (Y-u_\Lambda)/\Lambda_r$ conserve les dimensions des k_Λ-espaces vectoriels $(B_{\Lambda,r})_l$ pour tout $l > 1$. Par conséquent $\tilde{I}_{\Lambda,i}$ a la même dimension et le même degré que I_i.

Une relation $P_u(X_0,\dots,X_r,u_\Lambda) \in I_{\Lambda,i}$ de dépendance entière de u_Λ sur $A_{\Lambda,r}$ donne une relation de dépendance entière $P_u(X_0,\dots,X_{r-1},(Y-u_\Lambda)/\Lambda_r,u_\Lambda) \in \tilde{I}_{\Lambda,i}$ de u_Λ sur $\tilde{A}_{\Lambda,r}$ modulo $\tilde{I}_{\Lambda,i}$. Ensuite une relation $P_j(X_0,\dots,X_r,X_j) \in I_i$ de dépendance entière de X_j, pour $j \ge r+1$, sur A_r donne une relation de dépendance entière $P_j(X_0,\dots,X_{r-1},(Y-u_\Lambda)/\Lambda_r,X_j) \in \tilde{I}_{\Lambda,i}$ de X_j sur $k_\Lambda[X_0,\dots,X_{r-1},Y,u_\Lambda]$. La position de Noether de $\tilde{I}_{\Lambda,i}$ découle alors de la Proposition 27.2. En combinant à nouveau les Théorèmes 26.5 et 26.7 pour la suite régulière $\tilde{f}_{\Lambda,1},\dots,\tilde{f}_{\Lambda,i}$, on obtient que la dimension de $\tilde{E}_{\Lambda,r}$ est encore δ_i.

Du Théorème 27.16 on a l'inclusion

$$(q(u),q'(u)X_{r+1} - w_{r+1}(u),\dots,q'(u)X_n - w_n(u)) \subseteq I_i.$$

Comme le discriminant de q en T est non nul, le discriminant de q_Λ en T est non nul (en utilisant la Proposition 6.8 avec la spécialisation $\Lambda_{r+1} = \lambda_{r+1},\dots,\Lambda_n = \lambda_n$). Le même argument donne aussi que $q'(T)|_{X_r \mapsto (Y-T)/\Lambda_r}$ est inversible modulo $\tilde{q}_\Lambda$, et on obtient ainsi

$$(\tilde{q}_\Lambda(u_\Lambda),X_{r+1} - \tilde{v}_{\Lambda,r+1}(u_\Lambda),\dots,X_n - \tilde{v}_{\Lambda,n}(u_\Lambda)) \subseteq \tilde{I}_{\Lambda,i}.$$

Comme la dimension de $\tilde{\mathrm{E}}_{\Lambda,r}$ coïncide avec $\deg q$, on obtient que u_Λ est primitif pour $\tilde{\mathrm{I}}'_{\Lambda,i}$ et que la paramétrisation associée donne l'isomorphisme

$$\begin{aligned} \tilde{\mathrm{E}}_{\Lambda,r} &\to \tilde{\mathrm{K}}_{\Lambda,r}[\mathrm{T}]/(\tilde{q}_\Lambda(\mathrm{T})) \\ \mathrm{X}_j &\mapsto \tilde{v}_{\Lambda,j}(\mathrm{T}), \quad \text{pour } r+1 \le j \le n. \end{aligned}$$

Le Lemme 6.9 implique que Q_Λ est le coefficient constant du polynôme caractéristique de multiplication par $\tilde{f}_{\Lambda,i+1}$ dans $\tilde{\mathrm{E}}_{\Lambda,r}$. En particulier Q_Λ est un polynôme homogène de $k_\Lambda[\mathrm{X}_0,\dots,\mathrm{X}_{r-1},\mathrm{Y}]$ de degré total $d_{i+1}\delta_i = \delta_{i+1}$, par la Proposition 27.9. Le Corollaire 27.14 donne $\mathrm{Q}_\Lambda(\mathrm{Y}) \in \tilde{\mathrm{I}}_{\Lambda,i+1}$. Par le Lemme 27.11, on a aussi que Q_Λ est unitaire un Y. En remplaçant Y par $\Lambda_r \mathrm{X}_r + u_\Lambda$ on obtient finalement que Q_Λ est le polynôme minimal de U_Λ modulo $\mathrm{I}'_{\Lambda,i+1}$. La dernière inclusion provient du Lemme 27.15. ■

Calcul du résultant

La Proposition 28.3 prouve que le résultant calculé est un polynôme en Y dont on connaît le degré, mais, comme dans l'Exemple 28.8, les polynômes dont on calcule le résultant ont pour coefficients des fractions rationnelles en Y qui ne sont généralement pas des polynômes. Le calcul avec ces fractions peut mener à des fractions de numérateurs et dénominateurs de hauts degrés qui ne se simplifient qu'en fin de calcul, menant à une complexité élevée. Le calcul rapide par évaluation-interpolation n'est pas non plus une solution immédiate, puisqu'il faut éviter les valeurs qui annulent un dénominateur. La solution adoptée dans l'Algorithme 28.3, qui résume la méthode pour calculer la nouvelle paramétrisation, est de calculer ce résultant en série à précision plus grande que le degré, au voisinage d'un point b donné en argument.

Dans l'algorithme, la spécialisation des variables $\Lambda_1,\dots,\Lambda_n$ en les valeurs $\lambda_1,\dots,\lambda_n$ est notée $\Lambda_{1:n} \mapsto \lambda_{1:n}$.

Spécialisations

Nous dirons qu'un algorithme *commute avec la spécialisation d'une variable* X en la valeur a si la spécialisation de la sortie de l'algorithme coïncide avec la sortie de l'algorithme appliqué aux entrées spécialisées. Pour simplifier les analyses de complexité on utilise volontairement la notation $\tilde{\mathrm{O}}$ (Section 1.2). On rappelle qu'en degré d des polynômes à une variable sur un anneau peuvent être multipliés et divisés en coûts quasi-linéaires (Chapitres 2 et 3).

Les algorithmes de ce chapitre dépendent de plusieurs choix aléatoires. D'une part la mise en position de Noether, incluse dans les hypothèses pour simplifier la présentation, sous-entend qu'un changement de variable linéaire pris au hasard a déjà été appliqué. Ensuite, nous utilisons explicitement un point $(\lambda_1,\dots,\lambda_n)$ de k^n pour construire les éléments primitifs, et un autre point $(a_0,\dots,a_{n-1}) \in k^n$ pour spécialiser les variables libres dans la prochaine section. Sur le même plan, un point auxiliaire $b \in k$ intervient pour effectuer des calculs de résultants rapides de polynômes à deux variables.

Entrée $f_1,\dots,f_{i+1}$ satisfaisant les hypothèses (H_1)–(H_3), $(\lambda_1,\dots,\lambda_n)\in k^n$, $b\in k$, et la paramétrisation sous forme de Kronecker $q,w_{r+1},\dots,w_n$ de I_i' par l'élément primitif $u=\lambda_{r+1}X_{r+1}+\cdots+\lambda_nX_n$.

Sortie La paramétrisation sous forme de Kronecker de I_{i+1} par l'élément primitif $U=\lambda_rX_r+\cdots+\lambda_nX_n$.

1. Calculer la paramétrisation $q_\Lambda, v_{\Lambda,r+1},\dots,v_{\Lambda,n}$ de $I_{\Lambda,i}$ pour l'élément primitif $u_\Lambda=\Lambda_{r+1}X_{r+1}+\cdots+\Lambda_{r+1}X_{r+1}$ modulo $\mathfrak{L}^2$ où $\mathfrak{L}=(\Lambda_1-\lambda_1,\dots,\Lambda_n-\lambda_n)$:
   - $q_\Lambda(T)=q(T)-\sum_{i=r+1}^n(\Lambda_i-\lambda_i)w_i(T) \bmod \mathfrak{L}^2$;
   - $\Delta(T)=(q_\Lambda(T)-q(T))/q'(T) \bmod q(T) \bmod \mathfrak{L}^2$;
   - pour $j=r+1,\dots,n$,
     - $v_j(T)=w_j(T)/q'(T) \bmod q(T)$;
     - $v_{\Lambda,j}(T)=v_j(T)+(v_j'(T)\Delta(T) \bmod q(T)) \bmod \mathfrak{L}^2$,
     - $w_{\Lambda,j}(T)=q_\Lambda'(T)v_{\Lambda,j}(T) \bmod q_\Lambda(T) \bmod \mathfrak{L}^2$.
2. Effectuer le changement de variable mod $\mathfrak{L}^2$ et $(Y-b)^{\delta_{i+1}+1}$:
   - $\tilde q_\Lambda(T)=q_\Lambda(T)|_{X_r\mapsto(Y-T)/\Lambda_r}$;
   - $\tilde p_\Lambda(T)=(q_\Lambda'(T)\big|_{X_r\mapsto(Y-T)/\Lambda_r})^{-1} \bmod \tilde q_\Lambda(T)$;
   - pour $j=r+1,\dots,n$,
     - $\tilde w_{\Lambda,j}(T)=w_{\Lambda,j}(T)\big|_{X_r\mapsto(Y-T)/\Lambda_r}$;
     - $\tilde v_{\Lambda,j}(T)=\tilde p_\Lambda(T)\tilde w_{\Lambda,j}(T) \bmod \tilde q_\Lambda(T)$.
3. Calculer le résultant :
   - $\tilde h(T)=f_{i+1}(X_0,\dots,X_{r-1},(Y-T)/\Lambda_r,\tilde v_{\Lambda,r+1}(T),\dots,\tilde v_{\Lambda,n}(T))$ modulo $q_\Lambda(T)$, $\mathfrak{L}^2$ et $(Y-b)^{\delta_{i+1}+1}$;
   - $Q_\Lambda(Y)=\mathrm{Res}_T(\tilde q_\Lambda(T),\tilde h(T)) \bmod \mathfrak{L}^2 \bmod (Y-b)^{\delta_{i+1}+1}$.
4. Renvoyer
$$Q_\Lambda(T)|_{\Lambda_{1:n}\mapsto\lambda_{1:n}},-\frac{\partial Q_\Lambda(T)}{\partial\Lambda_r}\Big|_{\Lambda_{1:n}\mapsto\lambda_{1:n}},\dots,-\frac{\partial Q_\Lambda(T)}{\partial\Lambda_n}\Big|_{\Lambda_{1:n}\mapsto\lambda_{1:n}}.$$

Algorithme 28.3 – Calcul de la paramétrisation suivante.

Proposition 28.4 Sous les hypothèses (H_1)–(H_3), il existe une variété algébrique propre Z_i de $\mathbb{A}^n$ telle que pour tout $(\lambda_1,\dots,\lambda_n)$ pris hors de Z_i, il existe un sous-ensemble fini $Z_{\lambda,i}$ de k tel que pour tout b pris hors de $Z_{\lambda,i}$, on a :
- l'Algorithme 28.3 est correct ;
- il existe une variété algébrique propre $Z_{\lambda,b,i}$ de $\mathbb{A}^r$ telle que l'algorithme commute avec la spécialisation des variables $X_0,\dots,X_{r-1}$ en $a_0,\dots,a_{r-1}$ pris hors de $Z_{\lambda,b,i}$. Dans ce cas, le calcul spécialisé utilise $(nL+n^2)\tilde O(\delta_i\delta_{i+1})$ opérations dans k, où L est la complexité d'évaluation du DAG représentant $f_1,\dots,f_{i+1}$.

Démonstration. Si on oublie le « modulo $\mathfrak{L}^2$ », alors les étapes (1)–(3) suivent juste la

proposition précédente. En dérivant $Q_\Lambda(U_\Lambda) \in I_{i+1}^\Lambda$ en Λ_j on obtient

$$Q'_\Lambda(T)X_j + \frac{\partial Q_\Lambda(T)}{\partial \Lambda_j} \in I_{i+1}^\Lambda.$$

Le discriminant de $Q_\Lambda(T)$ est un polynôme de $k[\Lambda_r,\dots,\Lambda_n][X_0,\dots,X_r]$: on peut prendre n'importe quel coefficient non nul de ce discriminant pour définir une variété affine notée Z'_i. Pour tout point $(\lambda_1,\dots,\lambda_n)$ hors de Z'_i on peut spécialiser les Λ_j. En posant

$$Q(T) = Q_\Lambda(T)|_{\Lambda_{1:n}\mapsto\lambda_{1:n}}, \quad W_j(T) = \left.\frac{\partial Q_\Lambda(T)}{\partial \Lambda_r}\right|_{\Lambda_{1:n}\mapsto\lambda_{1:n}},$$

on obtient

$$(Q(U), Q'(U)X_r - W_r(U),\dots, Q'(U)X_n - W_n(U)) \subseteq I_{i+1}.$$

Comme Q est sans carré de degré δ_{i+1}, on obtient ainsi une paramétrisation sous forme de Kronecker de I'_{i+1}.

À ce stade, la variété Z'_i conviendrait à faire fonctionner l'algorithme si on utilisait des calculs de résultant sans division (par exemple en combinant sa définition sous forme de déterminant vue en Section 6.2 à l'algorithme de Berkowitz du Théorème 8.9). Mais ceci nous conduirait à une complexité bien plus élevée que souhaitée. Pour que l'algorithme soit efficace il doit utiliser l'algorithme d'Euclide rapide vu en Section 6.3, qui a un coût quasi-linéaire.

Une fois fixé un algorithme rapide pour le résultant, on peut prendre pour Z_i la réunion de Z'_i et d'une variété contenant les valeurs $(\lambda_1,\dots,\lambda_n)$ qui ne commutent pas avec la spécialisation $\Lambda_{1:n} \mapsto \lambda_{1:n}$ lors du calcul de $Q_\Lambda(Y) = \mathrm{Res}_T(\tilde{q}_\Lambda(T), \tilde{h}(T))$. Une fois $(\lambda_1,\dots,\lambda_n)$ fixés, on exclut ensuite les valeurs de b telles que ce calcul ne commute pas avec la spécialisation $Y \mapsto b$. Avec de bonnes valeurs des λ_i et de b ce calcul de résultant peut donc s'effectuer modulo $\mathfrak{L}^2$ et $(Y-b)^{\delta_{i+1}+1}$. Cette dernière précision est suffisante puisque Q(Y) est de degré δ_{i+1}. Sous ces conditions le résultat renvoyé par l'algorithme est correct.

Une fois $(\lambda_1,\dots,\lambda_n)$ et b fixés, on prend pour $Z_{\lambda,b,i}$ la variété définie par tous les dénominateurs et tests à zéro en les $X_0,\dots,X_{r-1}$ intervenant dans les calculs.

On analyse maintenant le coût de l'algorithme pour un calcul spécialisé. Les variables restant en jeu sont $X_r,\dots,X_n$, Y, T et les Λ_j. Tout d'abord, l'étape (1) requiert $n^2\tilde{O}(\delta_i^2)$ opérations dans k.

Les calculs de l'étape (2) peuvent aussi être réalisés en coût $n^2\tilde{O}(\delta_i^2)$. Car d'une part un changement de variable

$$P(X_r, T)|_{X_r\mapsto(Y-T)/\Lambda_r}$$

s'obtient comme

$$P((Y-T)/\lambda_r, T) - \frac{\partial P}{\partial X_r}((Y-T)/\lambda_r, T)(\Lambda_r - \lambda_r) \bmod \mathfrak{L}^2.$$

Et d'autre part, le calcul de $P((Y-T)/\lambda_{r+1}, T)$ se décompose en les calculs de $P_l((Y-1)/\lambda_{r+1}, 1)$ pour chaque composante homogène P_l de P. Il s'agit donc d'appliquer une

Entrée $f_1,\dots,f_n$ satisfaisant les hypothèses (H_1)–(H_3), $(\lambda_1,\dots,\lambda_n)\in k^n$, et $b\in k$.
Sortie La paramétrisation sous forme de Kronecker de I_n par l'élément primitif $\lambda_1X_1+\cdots+\lambda_nX_n$.

1. Construire la paramétrisation sous forme de Kronecker de I_1 par λ_nX_n comme $q(T)=f_1|_{X_n\mapsto T/\lambda_n}$ et $w_n(T)=q'(T)T/\lambda_n \bmod q(T)$.
2. Pour i de 1 à $n-1$, utiliser l'Algorithme 28.3 avec $f_1,\dots,f_{i+1}$, $(\lambda_1,\dots,\lambda_n)$ et b pour calculer la paramétrisation de I_{i+1} pour l'élément primitif $\lambda_rX_r+\cdots+\lambda_nX_n$, à partir de la paramétrisation de I_i.

Algorithme 28.4 – Calcul incrémental de paramétrisations.

translation et une dilatation à chaque $P_l(X_r,1)$. Cette opération requiert un coût quasi-linéaire, par exemple en utilisant les algorithmes d'évaluation et interpolation rapides du Chapitre 5. Au total, pour chaque polynôme P de degré total $\le\delta_i$, le changement de variables $X_r\mapsto(Y-T)/\Lambda_r$ requiert $\tilde{O}(\delta_i^2)$ opérations dans k. L'étape (2) accumule essentiellement $O(n^2)$ fois ce coût.

Pour l'étape (2), l'inversion modulaire pour obtenir $\tilde{p}_\Lambda$ requiert $n\tilde{O}(\delta_i\delta_{i+1})$ opérations dans k avec l'algorithme de résultant rapide. L'évaluation de f_{i+1} coûte ensuite $nL\tilde{O}(\delta_i\delta_{i+1})$. Le résultant pour obtenir Q_Λ fait aussi intervenir $n\tilde{O}(\delta_i\delta_{i+1})$ opérations dans k. La dernière étape de l'algorithme n'effectue pas vraiment de calcul. ■

Vue d'ensemble de l'approche incrémentale

La résolution incrémentale complète est détaillé dans l'Algorithme 28.4 : on calcule des paramétrisations successives pour I_1', I_2', I_3', etc.

Proposition 28.5 Sous les hypothèses (H_1)–(H_3), il existe une variété algébrique propre Z de $\mathbb{A}^n$ telle que pour tout $(\lambda_1,\dots,\lambda_n)$ pris hors de Z, il existe un sous-ensemble fini Z_λ tel que pour tout b pris hors de Z_λ l'Algorithme 28.4 est correct.

Démonstration. La preuve découle de la Proposition 28.4, en prenant pour Z la réunion des Z_i, et pour Z_λ la réunion des $Z_{\lambda,i}$. ■

En terme d'efficacité, l'Algorithme 28.4 est clairement pénalisé par les calculs avec les fractions rationnelles en les variables $X_1,\dots,X_{r-1}$ à l'étape i de la boucle. Le reste de ce chapitre consiste à raffiner cet algorithme pour s'affranchir de ces fractions rationnelles.

28.3 Remontée d'une paramétrisation

Pour contenir la croissance des expressions faisant intervenir un grand nombre de variables libres dans l'Algorithme 28.4, l'idée suivante est de construire incrémentalement non pas les paramétrisations des I_i' mais leurs spécialisations en les

variables libres. Plus précisément, on considère un point $a = (a_0, \dots, a_{n-1})$ de k^n, et la spécialisation d'une paramétrisation de I'_i où les variables $\mathrm{X}_0, \dots, \mathrm{X}_r$ sont respectivement remplacées par $a_0, \dots, a_r$. D'une façon plus compacte nous notons cette spécialisation $\mathrm{X}_{0:r} \mapsto a_{0:r}$. Nous avons vu dans la Proposition 27.17, que si a est choisi en dehors d'une variété algébrique propre de $\mathbb{A}^n$, alors l'idéal non homogène $\mathrm{I}_{a,i} = \mathrm{I}_i + (\mathrm{X}_0 - a_0, \dots, \mathrm{X}_r - a_r)$ est radical de dimension 0, et que

$$q_a(\mathrm{T}) = q(\mathrm{T})|_{\mathrm{X}_{0:r} \mapsto a_{0:r}}, w_{a,r+1}(\mathrm{T}) = w_{r+1}(\mathrm{T})|_{\mathrm{X}_{0:r} \mapsto a_{0:r}}, \dots, w_{a,n}(\mathrm{T}) = w_n(\mathrm{T})|_{\mathrm{X}_{0:r} \mapsto a_{0:r}}$$

constituent une paramétrisation de $\mathrm{I}_{a,i}$. Le problème que nous examinons est la remontée de la paramétrisation $q, w_{r+1}, \dots, w_n$ de I_i par l'élément primitif u à partir de sa spécialisation $q_a, w_{a,r+1}, \dots, w_{a,n}$. Implicitement, nous avons supposé que q_a est sans carré, et cela nous permet de quitter la représentation de Kronecker. Nous notons alors $v_j(\mathrm{T}) = w_j(\mathrm{T})/q'(\mathrm{T}) \bmod q(\mathrm{T})$, et $v_{a,j}(\mathrm{T}) = w_{a,j}(\mathrm{T})/q'_a(\mathrm{T}) \bmod q_a(\mathrm{T})$ pour $r+1 \le j \le n$.

Itération de Newton

La remontée repose sur un calcul de développement en séries de q et des v_j dans $\mathfrak{A}_{a,r} = k[[\mathrm{X}_0 - a_0, \dots, \mathrm{X}_r - a_r]]$. On note $\mathfrak{m}$ l'idéal maximal

$$\mathfrak{m} = (\mathrm{X}_0 - a_0, \dots, \mathrm{X}_r - a_r).$$

L'algorithme construit successivement des polynômes $q_{[m]} = q \bmod \mathfrak{m}^m$ et $v_{[m],j} = v_j \bmod \mathfrak{m}^m$, en partant de $q_{[1]} = q_a$ et $v_{[1],j} = v_{a,j}$, et en procédant récursivement par doublement de précision m.

L'idée principale est d'utiliser une itération de Newton pour le système $f_1 = \cdots = f_i = 0$ comme dans le Théorème 3.12 p. 70. On note

$$\mathrm{J} = \left(\partial \mathrm{F}_j / \partial \mathrm{X}_k\right)_{\substack{1 \le j \le i \\ r+1 \le k \le n}}$$

la matrice jacobienne de $f_1, \dots, f_i$ par rapport aux variables $\mathrm{X}_{r+1}, \dots, \mathrm{X}_n$. Pour garder les notations compactes dans l'opérateur de Newton, $v_{[m],r+1:n}$ représente le $(n-r)$-uplet $v_{[m],r+1}, \dots, v_{[m],n}$. On commence par calculer

$$\hat{v}_{[2m],r+1:n}(\mathrm{T}) = v_{[m],r+1:n}(\mathrm{T}) - \mathrm{J}^{-1} f_{1:i}\big|_{\mathrm{X}_{r+1:n} \mapsto v_{[m],r+1:n}(\mathrm{T})} \bmod q_{[m]}(\mathrm{T}) \bmod \mathfrak{m}^{2m},$$

ce qui est possible du fait du résultat suivant :

Lemme 28.6 Sous les hypothèses (H_1)–(H_3), si q_a est sans carré, alors la matrice jacobienne J de $(f_1, \dots, f_i)$ par rapport aux variables $\mathrm{X}_{r+1}, \dots, \mathrm{X}_n$ est inversible modulo $\mathrm{I}_{a,i}$.

Démonstration. On considère à nouveau le polynôme minimal q_Λ de $u_\Lambda = \Lambda_{r+1}\mathrm{X}_{r+1} + \cdots + \Lambda_n \mathrm{X}_n$. D'après le Lemme 27.15, le polynôme q_Λ est sans carré, et $q_\Lambda(u_\Lambda)$ appartient à I_i^Λ. Il existe donc des polynômes $g_1, \dots, g_i$ de R^Λ tels que

$$q_\Lambda(u_\Lambda) = g_1 f_1 + \cdots + g_i f_i.$$

En dérivant par rapport à X_j on obtient

$$\Lambda_j q'_\Lambda(u_\Lambda) = g_1 \frac{\partial f_1}{\partial X_j} + \cdots + g_i \frac{\partial f_i}{\partial X_j} \bmod I_i^\Lambda.$$

On peut spécialiser les Λ_j en les λ_j pour $r+1 \le j \le n$ et les X_j en les a_j pour $0 \le j \le r$:

$$\lambda_j q'_a(u) = g_1 \frac{\partial f_1}{\partial X_j} + \cdots + g_i \frac{\partial f_i}{\partial X_j} \bmod I_{a,i}.$$

Comme $q'_a(u)$ est inversible modulo $I_{a,i}$, pour tout choix de vecteur $\lambda = (\lambda_{r+1}, \ldots, \lambda_n)$ en dehors d'une variété propre de $\mathbb{A}^{n-r}$, ces équations fournissent un vecteur $g = (g_1, \ldots, g_i)$ tel que $J \cdot g = \lambda \bmod I_{a,i}$, ce qui prouve l'inversibilité de J modulo $I_{a,i}$. ■

Déformation du paramètre

Il existe un unique polynôme $\Delta(T) \in \mathfrak{R}_{a,r}[T]$ de degré $< \deg q$ et à coefficients dans $\mathfrak{m}$ tel que

$$q_{[2m]}(T + \Delta(T)) = 0 \bmod q_{[m]}(T) \bmod \mathfrak{m}^{2m},$$

qui est déterminé par le développement

$$q_{[2m]}(T + \Delta(T)) = q_{[2m]}(T) + q'_{[m]}(T)\Delta(T) \bmod q_{[m]}(T) \bmod \mathfrak{m}^{2m}.$$

Ensuite, pour $1 \le j \le i$, on déduit de $f_j(X_0, \ldots, X_r, v_{r+1}(T), \ldots, v_n(T)) = 0 \bmod q(T)$ que

$$f_j(X_0, \ldots, X_r, v_{[2m],r+1}(T + \Delta(T)), \ldots, v_{[2m],n}(T + \Delta(T))) = 0 \bmod q_{[m]}(T) \bmod \mathfrak{m}^{2m}.$$

Par unicité de la remontée par opérateur de Newton, on obtient alors

$$v_{[2m],j}(T + \Delta(T)) = \hat{v}_{[2m],j}(T) \bmod q_{[m]}(T) \bmod \mathfrak{m}^{2m}.$$

Ensuite, à partir de

$$u(v_{[2m],r+1}(T + \Delta(T)), \ldots, v_{[2m],n}(T + \Delta(T))) = T + \Delta(T) \bmod q_{[m]}(T) \bmod \mathfrak{m}^{2m}$$

on obtient ainsi une façon de calculer Δ :

$$\Delta(T) = u(\hat{v}_{[2m],r+1}(T), \ldots, \hat{v}_{[2m],n}(T)) - T.$$

Finalement on vient de voir, qu'à partir des $\hat{v}_{[2m],j}$, il est aisé d'en déduire Δ puis $q_{[2m]}, v_{[2m],r+1}, \ldots, v_{[2m],n}$:

$$q_{[2m]}(T) = q_{[m]}(T) - (q'_{[m]}(T)\Delta(T) \bmod q_{[m]}(T)) \bmod \mathfrak{m}^{2m},$$

$$v_{[2m],j}(T) = \hat{v}_{[2m],j}(T) - (v'_{[m],j}(T)\Delta(T) \bmod q_{[m]}(T)) \bmod \mathfrak{m}^{2m}.$$

Entrée $f_1,\dots,f_i$ satisfaisant les hypothèses (H_1)–(H_3), $(\lambda_1,\dots,\lambda_n)\in k^n$, $(a_0,\dots,a_r)\in k^r$, et la spécialisation de la paramétrisation $q_a, w_{a,r+1},\dots,w_{a,n}$ sous forme de Kronecker de I_i pour l'élément primitif $u=\lambda_{r+1}X_{r+1}+\cdots+\lambda_nX_n$.

Sortie La paramétrisation $q, w_{r+1},\dots,w_n$ de I_i sous forme de Kronecker pour u.

1. Initialiser $m=1$ et $q_{[1]}=q_a$, et $v_{[1],j}=(q_a')^{-1}w_{a,j} \bmod q_a$ pour $r+1\le j\le n$.
2. Tant que $m\le\delta_i$ faire :
   a. calculer $\hat v_{[2m],r+1:n}=v_{[m],r+1:n}-J^{-1}f_{1:i}\big|_{X_{r+1:n}\mapsto v_{[m],r+1:n}}$ modulo $q_{[m]}$ et $\mathfrak{m}^{2m}$;
   b. calculer $\Delta(T)=u(\hat v_{[2m],r+1}(T),\dots,\hat v_{[2m],n}(T))-T$;
   c. calculer $q_{[2m]}=q_{[m]}-(q_{[m]}'\Delta \bmod q_{[m]}) \bmod \mathfrak{m}^{2m}$;
   d. pour $r+1\le j\le n$, calculer $v_{[2m],j}=\hat v_{[2m],j}-(v_{[m],j}'\Delta(T) \bmod q_{[m]}) \bmod \mathfrak{m}^{2m}$;
   e. remplacer m par $2m$.
3. Renvoyer $q_{[m]}$, $q_{[m]}'v_{[m],r+1} \bmod q_{[m]} \bmod \mathfrak{m}^m$, …, $q_{[m]}'v_{[m],n} \bmod q_{[m]} \bmod \mathfrak{m}^m$, vus comme polynômes de $A_r[T]$.

Algorithme 28.5 – Remontée de Hensel d'une paramétrisation.

Fin de la remontée

Une fois obtenus $q_{[m]}, v_{[m],r+1},\dots,v_{[m],n}$ à la précision $m=\delta_i+1$, on obtient immédiatement q puisqu'il est de degré total δ_i. Ensuite on calcule w_j comme $q_{[m]}'v_{[m],j} \bmod q_{[m]} \bmod \mathfrak{m}^m$ puisqu'il est lui aussi de degré total δ_i. La méthode complète de remontée de Hensel se résume donc en l'Algorithme 28.5. Dans l'algorithme principal de la section suivante la remontée ne concerne que la dernière variable libre X_r. Les autres X_j pour $1\le j\le r-1$ restent spécialisées à a_j.

Proposition 28.7 Sous les hypothèses (H_1)–(H_3), l'Algorithme 28.5 est correct. L'algorithme commute avec les spécialisations $X_{0:r-1}\mapsto a_{0:r-1}$, tant que $a_{0:r}$ maintient q_a sans carré. Si $f_1,\dots,f_i$ est donné par un DAG de complexité d'évaluation L, alors le calcul spécialisé coûte $(nL+n^4)\tilde O(\delta_i^2)$ opérations dans k.

Démonstration. La correction découle de la discussion qui précède. On analyse maintenant le coût de l'algorithme sur des entrées spécialisées. Les variables en jeu sont donc $X_r,\dots,X_n$ et T. D'une façon générale, pour évaluer un polynôme $g(Z_1,\dots,Z_l)$ et ses dérivées partielles en un point $z_1,\dots,z_l$, il suffit d'évaluer $g(z_1+\epsilon_1,\dots,z_l+\epsilon_l)$ modulo $(\epsilon_1,\dots,\epsilon_l)^2$, ce qui implique un surcoût de $O(l)$ par rapport à la seule évaluation $g(z_1,\dots,z_l)$. Par conséquent, évaluer les f_j et leurs dérivées partielles en précision m requiert $nL\tilde O(m\delta_i)$ opérations dans k.

Pour inverser la matrice $J_{[2m]}(T)=J|_{X_{r+1:n}\mapsto v_{[2m],r+1:n}(T)} \bmod q_{[2m]}(T) \bmod \mathfrak{m}^{2m}$, on procède par récurrence, à partir de l'inverse $J_{[m]}^{-1}(T)$ de $J_{[m]}(T)$ à la précision $\mathfrak{m}^m$.

Entrée $f_1,\dots,f_n$, satisfaisant les hypothèses (H_1)–(H_3), et $(\lambda_1,\dots,\lambda_n)\in k^n$, $b\in k$, $(a_0,\dots,a_{n-1})\in k^n$.
Sortie La paramétrisation de I_n par $u=\lambda_1X_1+\cdots+\lambda_nX_n$ spécialisée en $X_0\mapsto a_0$.

1. Calculer la paramétrisation de I_1 par $u=\lambda_nX_n$ spécialisée en $X_{0:n-1}\mapsto a_{0:n-1}$, c'est-à-dire $q_a(T)=f_1|_{X_{0:n-1}\mapsto a_{0:n-1},\ X_n\mapsto T/\lambda_n}$, $w_{a,n-1}(T)=Tq_a'(T) \bmod q_a(T)$.
2. Pour i de 1 à $n-1$:
   a. utiliser l'Algorithme 28.5 avec $f_1,\dots,f_i$, $(\lambda_1,\dots,\lambda_n)$, et $(a_0,\dots,a_r)$ pour remonter la spécialisation de la paramétrisation de I_i en $X_{0:r-1}\mapsto a_{0:r-1}$;
   b. utiliser l'Algorithme 28.3 avec $f_1,\dots,f_{i+1}$, $(\lambda_1,\dots,\lambda_n)$, et b, pour en déduire la spécialisation de la paramétrisation de I_{i+1} en $X_{0:r-1}\mapsto a_{0:r-1}$.
3. Renvoyer la paramétrisation de I_n spécialisée en $X_0\mapsto a_0$.

Algorithme 28.6 – Algorithme de résolution géométrique.

Au départ, lorsque $m=1$, on utilise l'algorithme de Berkowitz, qui prend $n^4\tilde{O}(\delta_i)$ opérations dans k (Théorème 8.9 page 177). Ensuite on utilise la formule classique d'inversion par itération de Newton :

$$J_{[2m]}^{-1}(T)=J_{[m]}^{-1}(T)-J_{[m]}^{-1}(T)(J_{[2m]}(T)J_{[m]}^{-1}(T)-Id) \bmod q_{[2m]}(T) \bmod \mathfrak{m}^{2m},$$

qui demande $n^3\tilde{O}(m\delta_i)$ opérations dans k. Notons que pour le calcul de l'étape (2.a) il est suffisant de connaître $J_{[m]}^{-1}(T)$.

En précision $2m$, les autres calculs demandent $n\tilde{O}(m\delta_i)$ opérations additionnelles dans k. Le coût total s'obtient de la même manière que pour les analyses de complexité similaires du Chapitre 3 : comme la précision double à chaque étape, le coût total est essentiellement dominé par le coût de la dernière étape. ■

28.4 Algorithme principal

En appliquant successivement étapes de remontée et d'ajout d'une nouvelle équation, on parvient finalement à un algorithme travaillant sur les spécialisations de paramétrisations. La méthode est synthétisée dans l'Algorithme 28.6.

Théorème 28.8 Soit k un corps de caractéristique zéro, et soient $f_1,\dots,f_n$ des polynômes homogènes de R, de degrés respectifs $d_i\geq 2$, et satisfaisant (H_1)–(H_3). Il existe une variété algébrique propre Z de $\mathbb{A}^n$, telle que pour tout point $(\lambda_1,\dots,\lambda_n)$ hors de Z, il existe un sous-ensemble fini Z_λ de k tel que pour tout b pris hors de Z_λ, il existe une variété algébrique propre $Z_{\lambda,b}$ de $\mathbb{A}^n$ telle que pour tout point $(a_0,\dots,a_{n-1})$ hors de $Z_{\lambda,b}$, l'Algorithme 28.6 renvoye un résultat correct, et utilise $(nL+n^4)\tilde{O}(\delta_n^2)$ opérations dans k, où $\delta_n=d_1\cdots d_n$.

Démonstration. Tout d'abord on choisit pour Z et Z_λ ceux de la Proposition 28.5. Un fois les λ_j et b fixés, l'Algorithme 28.4 s'exécute correctement. Les bonnes valeurs des a_j permettent de faire fonctionner l'Algorithme 28.3 correctement selon la Proposition 28.4. On adjoint aux mauvaises valeurs de $(a_0,\dots,a_{n-1})$ celles qui empêchent le bon fonctionnement de l'Algorithme 28.5 de remontée. Ceci détermine $Z_{\lambda,b}$. Avec ces choix, la correction de l'algorithme et son coût sont des conséquences des propositions 28.4 et 28.7. ■

Comparaison avec les bases de Gröbner

Arrivé à la fin de cette partie, nous disposons de deux méthodes, et d'estimations de complexité fournies par le théorème ci-dessus et par le Théorème 26.15 et son Corollaire 26.16. Il est naturel de souhaiter comparer ces résultats. Il faut cependant prendre des précautions : tout d'abord on ne compare que des bornes supérieures de complexité, et non la complexité réelle du calcul. Ensuite, ces deux résultats de complexité ne sont pas établis dans le même modèle de calcul : dans le cas des bases standard, le résultat est toujours correct et le système de polynômes est donné en représentation dense, alors que pour la résolution géométrique le comportement est probabiliste mais on peut tirer parti d'un DAG calculant efficacement les polynômes en entrée. Il faut donc garder à l'esprit que, si d est une borne sur les degrés de $f_1,\dots,f_n$, la complexité d'évaluation L peut varier de $O(1)$ à s fois le nombre $\binom{n+d}{d}$ de monômes homogènes de degré d en $n+1$ variables.

Une fois ceci posé, on peut se placer dans la situation simplifiée où les degrés des polynômes $(f_1,\dots,f_n)$ sont tous égaux à un $d \geq 2$ fixé, et considérer les logarithmes des bornes de complexité pour $n \to \infty$. L'estimation de complexité des bases standard de la Section 26.6 donne

$$\theta n\,(d\log d - (d-1)\log(d-1)) + O(\log n)$$

qu'il s'agit de comparer à celle du théorème précédent

$$2\log\delta_n + \log \mathrm{L} + O(\log n) = 2n\log d + O(\log n),$$

puisque $\log\binom{n+d}{d} = d\log n + O(1)$. Les ordres de grandeurs sont donc les mêmes, mais dans ce régime asymptotique, cette seconde estimation est légèrement inférieure à la première, même si $\theta = 2$, comme on le voit en utilisant $\log(d-1) < \log d$.

Test à zéro des DAG

Le Lemme 28.1 donne une condition pour garantir qu'un résultat calculé par l'algorithme de résolution géométrique est correct. Si la condition n'est pas satisfaite, alors on peut soupçonner la suite des polynômes en entrée de ne pas être régulière réduite, avec une forte probabilité. Il est possible de borner cette probabilité, et renvoyons le lecteur aux notes de fin de chapitre. Il est bien sûr possible de vérifier les hypothèses avec des calculs de base standard, mais avec un coût qui dépasse celui de l'algorithme de résolution géométrique. Nous abordons ici une facette du problème, qui soulève à l'heure actuelle toujours d'importantes questions théoriques.

Pour savoir si $f_1,\dots,f_n$ est une suite régulière, la première question à se poser est de savoir si f_1 est nul ou non. Afin de simplifier un peu la présentation, nous

avons considéré connus les degrés des polynômes. Néanmoins déterminer le degré d'un polynôme représenté par un DAG n'est pas une tâche aisée. En fait, sous réserve de connaître une borne sur le degré, elle se ramène au test à zéro d'un autre DAG. Pour déterminer si f_2 est diviseur de zéro ou non modulo (f_1), sous réserve d'avoir la position de Noether pour (f_1), on est ramené à tester si le résultant $\mathrm{Res}_{X_n}(f_1, f_2)$ est nul ou non (voir le Corollaire 27.14). Si f_1 et f_2 sont donnés par un DAG de complexité L et que les degrés d_1 et d_2 sont connus, alors ce résultant peut être représenté par un DAG de complexité $\tilde{\mathrm{LO}}(d_1+d_2)+\mathrm{O}((d_1+d_2)^4)$ en utilisant l'algorithme de Berkowitz (Théorème 8.9). Et on est ainsi ramené au test à zéro d'un DAG de complexité qui dépend de L et des degrés d_i. Plus généralement, on peut penser qu'on exécute l'algorithme de résolution géométrique en conservant les λ_j et les a_j comme des variables et en utilisant des DAG pour représenter tous les polynômes. On peut alors montrer que la vérification des hypothèses (H_1)–(H_3) se ramène à des tests à zéro de DAG de tailles $\mathrm{L}(n\delta_n)^{\mathrm{O}(1)}$.

Définition 28.2 Soient $\mathcal{P} \subseteq k[X_1,\dots,X_n]$ un ensemble de polynômes et $\mathcal{Q} \subseteq k^n$. On dit que l'ensemble $\mathcal{Q}$ est *questeur* pour $\mathcal{P}$ si aucun polynôme de $\mathcal{P}$ non nul ne s'annule identiquement sur $\mathcal{Q}$. Ceci revient à dire que l'évaluation sur $\mathcal{Q}$ est un test à zéro pour les polynômes de $\mathcal{P}$.

Ainsi, la question du test à zéro des DAG est résolue si on peut trouver des ensembles questeurs de taille convenable. Une première famille d'ensembles questeurs, dont la taille est malheureusement exponentielle en n, est donnée par le résultat suivant.

Proposition 28.9 Soient $\mathcal{E}_1,\dots,\mathcal{E}_n \subseteq k$ des parties de cardinal supérieur à $d+1$. Alors $\mathcal{Q} = \mathcal{E}_1 \times \cdots \times \mathcal{E}_n \subseteq k^n$ est un ensemble questeur pour les polynômes de $k[X_1,\dots,X_n]$ de degrés au plus d.

Démonstration. Procédons par récurrence sur n. Pour $n=1$, un polynôme de degré au plus d qui admet $d+1$ racines est nul. Remarquons que ceci est vrai généralement pour les polynômes à coefficients dans un anneau unitaire car la démonstration repose sur la division euclidienne par $X-x$ si x est une racine. Supposons le résultat vrai au rang $n-1$. Écrivons le $p = \sum p_i X_n^i$ avec $p_i \in k[X_1,\dots,X_{n-1}]$. Soit $x \in \mathcal{E}_n$, alors $p(X_1,\dots,X_{n-1},x)$ est nul car il s'annule sur $\mathcal{E}_1 \times \cdots \times \mathcal{E}_{n-1}$. Le polynôme p a au moins $d+1$ racines en X_n donc il est nul. ∎

Pour des polynômes creux (c'est-à-dire dont peu de monômes ont un coefficient non nul), on peut trouver d'autres ensembles questeurs plus adaptés. Soit $\mathcal{A} \subseteq \mathbb{N}^n$, et considérons $\mathcal{W}_{\mathcal{A}} = \{\sum_{a\in\mathcal{A}} c_a X^a \mid c_a \in k\}$, l'ensemble des polynômes de $k[X_1,\dots,X_n]$ à supports dans $\mathcal{A}$. On a le résultat suivant que nous ne prouvons pas :

Proposition 28.10 L'ensemble $2^{\{\mathcal{A}\}} = \{(2^{a_1},\dots,2^{a_n}) \mid (a_1,\dots,a_n) \in \mathcal{A}\}$ est questeur pour $\mathcal{W}_{\mathcal{A}}$.

Par ailleurs, mentionnons qu'il est conjecturé que $\mathcal{A}$ est un ensemble questeur pour

$\mathcal{W}_{\mathcal{A}}$. Pour conclure, mentionnons qu'il existe des ensembles questeurs de tailles polynomiales. Cependant on ne sait pas les construire efficacement d'une façon déterministe.

Proposition 28.11 Soit $\mathcal{W}(D, n, L)$ l'ensemble des polynômes en n variables de degré au plus D et de complexité d'évaluation au plus L. Fixons $\Gamma \subset k$ de cardinal $2L(D+1)^2$ et $m = 6(L+1)(L+n+1)$. Soient $\tau(D, n, L, \Gamma)$ l'ensemble des m-uplets de n-uplets d'éléments de Γ et $\xi(D, n, L, \Gamma)$ celui des éléments de $\tau(D, n, L, \Gamma)$ qui ne sont pas questeurs pour $\mathcal{W}(D, n, L)$. Alors on a :

$$\frac{|\xi(D, n, L, \Gamma)|}{|\Gamma|^{nm}} \leq \frac{1}{|\Gamma|^{m/6}}.$$

Notons que le membre droit de cette dernière inégalité est majoré par 1/264000 dans les cas non triviaux. Les algorithmes utilisant des DAG peuvent alors faire intervenir deux types de complexité. D'une façon informelle, la *complexité non uniforme* revient à dire, « au prix d'un précalcul, la complexité est ... ». Ce précalcul revient dans le cas présent à trouver un ensemble questeur. L'autre complexité, dite probabiliste, revient à dire « si on choisit un ensemble questeur au hasard ... ». Des études détaillées de ce aspects se trouvent dans les travaux mentionnés ci-dessous.

Notes

L'algorithme de résolution géométrique est aussi appelé parfois l'algorithme *Kronecker* en hommage aux travaux de Kronecker sur la résolution des systèmes algébriques [Kro82]. Ce chapitre simplifie une version déjà simplifiée présentée dans l'article de Durvye et Lecerf [DL07]. Une présentation plus détaillée dans le cas affine, comprenant des exemples et des illustrations, se trouve aussi dans la thèse de Durvye [Dur08]. L'analyse de complexité reprend celle de l'article de Giusti, Lecerf et Salvy [GLS01]. Une implémentation en C++ est disponible dans la librairie GEOMSOLVEX de MATHEMAGIX. Les résultats présentés dans ce chapitre sont en fait le fruit d'une série de travaux initiés par Giusti, Heintz, Morais et Pardo au début des années 90 [FGS95 ; GH93 ; GHS93 ; Giu+95 ; Giu+97 ; Giu+98 ; Hei+00 ; HMW01 ; KP96]. Une présentation historique est donnée dans l'introduction de l'article de Durvye et Lecerf [DL07].

Dans le cas où $k = \mathbb{Q}$ et $f_1, \ldots, f_n$ une suite régulière réduite, une bonne stratégie de calcul consiste à appliquer d'abord l'algorithme de résolution géométrique modulo un nombre premier suffisamment grand pour ne pas annuler les dénominateurs des nombres intervenant lors du calcul sur $\mathbb{Q}$, mais pas trop grand non plus pour que l'arithmétique modulaire soit rapide. Ensuite il est possible d'adapter l'algorithme de remontée de Hensel pour remonter des entiers p-adiques et reconstruire la résolution géométrique sur $\mathbb{Q}$. Nous renvoyons à l'article de Giusti, Lecerf et Salvy [GLS01] ainsi qu'à la thèse de Schost [Sch01] pour plus de détails.

Si la suite n'est pas régulière, l'ensemble des solutions du système doit être décomposé en une union de variétés de différentes dimensions. On parle alors de *décomposition équidimensionelle*. Si la suite n'est pas réduite, alors il faut utiliser des

généralisations de la méthode de Newton en présence de racines multiples, comme par exemple des techniques dites de *déflation*. Plusieurs articles traitent de ces questions [Jer+04 ; JPS01 ; JS00 ; JS02 ; Lec00 ; Lec02 ; Lec03]. Une courte description de ces travaux de généralisation se trouve aussi dans l'introduction de l'article de Durvye et Lecerf [DL07].

La preuve originale de la Proposition 28.10 se trouve dans l'article de Risler et Ronga [RR90]. Celle de la Proposition 28.11 peut être trouvée dans l'article de Heintz et Schnorr [HS82].

Bibliographie

DL07 Durvye, Clémence et Grégoire Lecerf (2007). « A concise proof of the Kronecker polynomial system solver from scratch ». In : *Expositiones Mathematicae*, vol. 26, n°2, p. 101–139.

Dur08 Durvye, Clémence (2008). « Algorithmes pour la décomposition primaire des idéaux polynomiaux de dimension nulle donnés en évaluation ». Thèse de doctorat. Université de Versailles Saint-Quentin-en-Yvelines.

FGS95 Fitchas, N., M. Giusti et F. Smietanski (1995). « Sur la complexité du théorème des zéros ». In : *Approximation and optimization in the Caribbean, II (Havana, 1993)*. Vol. 8. Approximation & optimization. Frankfurt am Main : P. Lang, p. 274–329.

GH93 Giusti, Marc et Joos Heintz (1993). « La détermination des points isolés et de la dimension d'une variété algébrique peut se faire en temps polynomial ». In : *Computational algebraic geometry and commutative algebra (Cortona, 1991)*. Vol. XXXIV. Sympos. Math. Cambridge : Cambridge Univ. Press, p. 216–256.

GHS93 Giusti, Marc, Joos Heintz et Juan Sabia (1993). « On the efficiency of effective Nullstellensätze ». In : *Computational Complexity*, vol. 3, n°1, p. 56–95.

Giu+95 Giusti, M., J. Heintz, J. E. Morais et L. M. Pardo (1995). « When polynomial equation systems can be "solved" fast ? » In : *Applied algebra, algebraic algorithms and error-correcting codes (Paris, 1995)*. Vol. 948. Lecture Notes in Computer Science. Springer-Verlag, p. 205–231.

Giu+97 Giusti, M., K. Hägele, J. Heintz, J. L. Montaña, J. E. Morais et L. M. Pardo (1997). « Lower bounds for Diophantine approximations ». In : *Journal of Pure and Applied Algebra*, vol. 117/118, p. 277–317.

Giu+98 Giusti, M., J. Heintz, J. E. Morais, J. Morgenstern et L. M. Pardo (1998). « Straight-line programs in geometric elimination theory ». In : *Journal of Pure and Applied Algebra*, vol. 124, n°1-3, p. 101–146.

GLS01 Giusti, Marc, Grégoire Lecerf et Bruno Salvy (2001). « A Gröbner free alternative for polynomial system solving ». In : *Journal of Complexity*, vol. 17, n°1, p. 154–211.

Hei+00 Heintz, Joos, Teresa Krick, Susana Puddu, Juan Sabia et Ariel Waissbein (2000). « Deformation techniques for efficient polynomial equation solving ». In : *Journal of Complexity*, vol. 16, n°1, p. 70–109.

HMW01 Heintz, Joos, Guillermo Matera et Ariel Waissbein (2001). « On the time–space complexity of geometric elimination procedures ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 11, n°4, p. 239–296.

HS82 Heintz, J. et C.-P. Schnorr (1982). « Testing polynomials which are easy to compute ». In : *Logic and algorithmic (Zurich, 1980)*. Vol. 30. Monograph. Enseign. Math. Geneva : Univ. Genève, p. 237–254.

Jer+04 Jeronimo, Gabriela, Teresa Krick, Juan Sabia et Martín Sombra (2004). « The computational complexity of the Chow form ». In : *Foundations of Computational Mathematics*, vol. 4, n°1, p. 41–117.

JPS01 Jeronimo, Gabriela, Susana Puddu et Juan Sabia (2001). « Computing Chow forms and some applications ». In : *Journal of Algorithms*, vol. 41, n°1, p. 52–68.

JS00 Jeronimo, Gabriela et Juan Sabia (2000). « Probabilistic equidimensional decomposition ». In : *Comptes Rendus des Séances de l'Académie des Sciences. Série I. Mathématique*, vol. 331, n°6, p. 485–490.

JS02 — (2002). « Effective equidimensional decomposition of affine varieties ». In : *Journal of Pure and Applied Algebra*, vol. 169, n°2-3, p. 229–248.

KP96 Krick, T. et L. M. Pardo (1996). « A computational method for Diophantine approximation ». In : *Algorithms in algebraic geometry and applications (Santander, 1994)*. Vol. 143. Progress in Mathematics. Birkhäuser, p. 193–253.

Kro82 Kronecker, Leopold (1882). « Grundzüge einer arithmetischen Theorie der algebraischen Grössen ». In : *Journal für die Reine und Angewandte Mathematik*, vol. 92, p. 1–122.

Lec00 Lecerf, Grégoire (2000). « Computing an equidimensional decomposition of an algebraic variety by means of geometric resolutions ». In : *ISSAC'00 : International Symposium on Symbolic and Algebraic Computation*. ACM, p. 209–216.

Lec02 — (2002). « Quadratic Newton iteration for systems with multiplicity ». In : *Foundations of Computational Mathematics*, vol. 2, n°3, p. 247–293.

Lec03 — (2003). « Computing the equidimensional decomposition of an algebraic closed set by means of lifting fibers ». In : *Journal of Complexity*, vol. 19, n°4, p. 564–596.

RR90 Risler, Jean-Jacques et Felice Ronga (1990). « Testing polynomials ». In : *Journal of Symbolic Computation*, vol. 10, n°1, p. 1–5.

Sch01 Schost, Éric (2001). « Sur la résolution des systèmes polynomiaux à paramètres ». Thèse de doctorat. Palaiseau, France : École polytechnique.

VI Sommation et intégration de suites et fonctions spéciales

29. Sommation hypergéométrique et solutions hypergéométriques de récurrences

Résumé

Le Chapitre 16 a indiqué comment un petit noyau d'algorithmes relativement simples permet de trouver les solutions polynomiales et rationnelles de récurrences linéaires. Nous appliquons ici tout cet ensemble d'algorithmes au calcul de sommes indéfinies et définies de suites hypergéométriques et à la recherche de solutions hypergéométriques de récurrences linéaires. La suite de cet ouvrage prolongera les méthodes de ce chapitre aux questions de sommation et d'intégration de suites et fonctions spéciales plus générales, solutions de systèmes d'équations différentielles et de relations de récurrence d'ordre quelconque.

Voici deux sommes classiques, la première due à Gauss, la seconde à Dixon, dont le traitement algorithmique est détaillé dans ce chapitre :

$$\sum_{k=0}^{\infty} \frac{(a)_k (b)_k}{(c)_k k!} = \frac{\Gamma(c-a-b)\Gamma(c)}{\Gamma(c-a)\Gamma(c-b)},$$

$$\sum_{k\in\mathbb{Z}} (-1)^k \binom{a+b}{a+k}\binom{a+c}{c+k}\binom{b+c}{b+k} = \frac{(a+b+c)!}{a!\,b!\,c!}.$$

Dans la première identité, valide pour $\mathrm{Re}(c-a-b) > 0$ pour assurer la convergence de la série, $(x)_n$ représente le produit $x(x-1)\cdots(x-n+1)$ et $\Gamma(s)$ est la fonction obtenue par prolongement analytique à partir de la formule

$$\Gamma(x) = \int_0^{\infty} t^{x-1} e^{-t}\, dt \qquad \text{pour } \mathrm{Re}\, x > 0.$$

Pour la seconde identité, on adopte la convention que les coefficients binomiaux $\binom{n}{m}$ sont nuls lorsque $m < 0$ ou $m > n$.

Pour justifier de telle sommes *définies* — où toutes les valeurs possibles de l'indice de sommation sont parcourues —, il serait agréable de pouvoir faire appel, comme dans le cas de l'intégration où l'on recherche une primitive, à une somme *indéfinie* — c'est-à-dire vue comme fonction de sa borne supérieure — et de voir la somme définie comme différence de deux évaluations. Un cadre simple serait celui où, tout comme les sommants dans ces deux exemples, une telle somme indéfinie serait *hypergéométrique* — solution d'une récurrence linéaire d'ordre 1, voir la Définition 29.1 ci-dessous. Cependant, pour nos deux exemples, il n'existe pas de somme indéfinie hypergéométrique, et c'est le cas général. Les deux sommes définies peuvent néanmoins être calculées automatiquement, si l'on prend la peine de rechercher une suite auxiliaire qui, elle, admet une somme indéfinie hypergéométrique.

Une chaîne complète d'algorithmes permettant de trouver les membres droits des identités ci-dessus est détaillée dans ce chapitre : l'algorithme de Gosper de la Section 29.1 décide de l'existence d'une somme indéfinie hypergéométrique pour un sommant hypergéométrique ; en modifiant convenablement un sommant hypergéométrique, l'algorithme de Zeilberger de la Section 29.2 trouve, quand il en existe une, une récurrence vérifiée par une somme hypergéométrique définie. Pour ne pas s'arrêter sur une récurrence et revenir à une forme explicite, l'algorithme de Petkovšek de la Section 29.3 obtient toutes les solutions d'une récurrence linéaire qui sont données comme combinaisons linéaires de termes hypergéométriques.

29.1 Sommation hypergéométrique indéfinie. Algorithme de Gosper

La recherche de formes closes pour la sommation est souvent naturellement posée en termes de suites hypergéométriques. Après quelques définitions et propriétés de ces suites, nous abordons leur sommation indéfinie.

Suites hypergéométriques

Définition 29.1 On dit qu'une suite (u_n) est *hypergéométrique* s'il existe deux polynômes non nuls P et Q tels que pour tout $n \in \mathbb{N}$,

$$\mathrm{Q}(n)u_{n+1} = \mathrm{P}(n)u_n. \tag{29.1}$$

Autrement dit, elle est polynomialement récurrente et vérifie une récurrence linéaire *d'ordre 1*.

Cette définition mérite quelques observations :

- u_{n+1} est totalement déterminé par u_n dès lors que $\mathrm{Q}(n) \neq 0$.
- Il n'est pas exigé que $\mathrm{pgcd}(\mathrm{P}, \mathrm{Q}) = 1$: en particulier, si un entier n_0 est zéro commun de P et Q, les termes u_{n_0} et u_{n_0+1} sont indépendants l'un de l'autre.
- Si P s'annule sur un entier n_0 sans que Q ne s'annule sur aucun entier $n \geq n_0$, alors $u_n = 0$ dès lors que $n > n_0$.

Exercice 29.1 — Vérifier que la suite définie pour tout $n \in \mathbb{N}$ par $u_n = \binom{5}{n}$ satisfait

pour tout $n \in \mathbb{N}$ la relation de récurrence

$$(n+1)u_{n+1} = (5-n)u_n. \tag{29.2}$$

— Considérer la suite définie pour $n \geq 6$ par $v_6 = 1$ et l'équation (29.2). Calculer v_7, v_8 et v_9.
— On prolonge v par $v_0 = \cdots = v_5 = 1$. Donner une relation de récurrence de la forme (29.1) montrant que v est hypergéométrique.
— Même question pour la suite w définie par $w_n = u_n$ si $n \leq 5$ et $w_n = v_n$ si $n \geq 6$.

■

La récurrence (29.1) étant d'ordre 1, il est facile d'en écrire la solution générale au moins lorsque Q n'a pas de racine entière positive :

$$u_n = u_0 \left(\frac{\mathrm{lc}(\mathrm{P})}{\mathrm{lc}(\mathrm{Q})}\right)^n \frac{\prod_{\mathrm{P}(\alpha)=0}(-\alpha)(-\alpha+1)\cdots(-\alpha+n-1)}{\prod_{\mathrm{Q}(\beta)=0}(-\beta)(-\beta+1)\cdots(-\beta+n-1)},$$

où $\mathrm{lc}(p)$ désigne le coefficient de tête du polynôme p et où chaque produit doit être compris avec répétition des racines selon leurs multiplicités. Ici, on peut sans perte de généralité supposer $\mathrm{pgcd}(\mathrm{P},\mathrm{Q}) = 1$.

Lorsque Q n'a pas de zéro dans $\mathbb{N}$, la solution (u_n) de l'équation (29.1) est totalement déterminée par la valeur u_0, qui joue le rôle de condition initiale. Sinon, la suite (u_n) est déterminée par ses valeurs en $n = 0$ et en les $n \in \mathbb{N}$ tels que $\mathrm{Q}(n-1) = 0$.

Les suites hypergéométriques jouent un rôle important en analyse classique, où elles apparaissent comme coefficients de Taylor des séries introduites par la définition suivante.

Définition 29.2 On appelle *série hypergéométrique généralisée* et on note

$${}_p\mathrm{F}_q\left(\begin{matrix}a_1,\ldots,a_p\\ b_1,\ldots,b_q\end{matrix}\middle| x\right)$$

la série

$$\sum_{n\geq 0} \frac{(a_1)_n\cdots(a_p)_n}{(b_1)_n\cdots(b_q)_n}\frac{x^n}{n!},$$

où la notation $(a)_n$ représente le produit $a(a-1)\cdots(a-n+1)$, appelé *symbole de Pochhammer*, qui s'exprime aussi comme $\Gamma(a+n)/\Gamma(a)$.

Des cas particuliers de séries qui s'expriment à l'aide de séries hypergéométriques généralisées sont les séries $\exp(x)$, $\log(1+x)$, les dilogarithmes et polylogarithmes, les fonctions $\mathrm{J}_\nu(x)$ de Bessel, les fonctions d'Airy $\mathrm{Ai}(x)$, etc. Ainsi :

$$\exp(x) = {}_0\mathrm{F}_0\left(\begin{matrix}-\\ -\end{matrix}\middle| x\right),$$

$$\log(1+x) = {}_2\mathrm{F}_1\left(\begin{matrix}1,1\\ 2\end{matrix}\middle| -x\right),$$

$$\mathrm{J}_\nu(x) = \left(\frac{x}{2}\right)^\nu \frac{1}{\Gamma(\nu+1)} {}_0\mathrm{F}_1\left(\begin{matrix}-\\ \nu+1\end{matrix}\middle| x\right),$$
$$\mathrm{Ai}(x) = -\frac{x\sqrt[6]{3}\Gamma(2/3)}{2\pi} {}_0\mathrm{F}_1\left(\begin{matrix}-\\ 4/3\end{matrix}\middle| \frac{x^3}{9}\right) + \frac{\sqrt[3]{3}}{3\Gamma(2/3)} {}_0\mathrm{F}_1\left(\begin{matrix}-\\ 2/3\end{matrix}\middle| \frac{x^3}{9}\right).$$

Exercice 29.2 Récrire l'identité de Dixon à l'aide de l'évaluation d'une série ${}_3\mathrm{F}_2$ en 1. ■

Terme hypergéométrique

La définition suivante permet de donner une idéalisation algébrique commode des suites numériques.

Définition 29.3 Un *anneau à différence* est un anneau commutatif A muni d'un endomorphisme d'anneau σ. Un *corps à différence* est un anneau à différence qui est un corps.

Exemple 29.1 Un exemple simple est le corps usuel de fractions rationnelles $\mathbb{Q}(g)$ muni du morphisme donné par $\sigma(g) = g+1$: le terme g s'identifie à la suite de terme général n. Cet exemple se prolonge sur le corps $\mathbb{Q}(g,h)$ des fractions rationnelles en deux variables, en étendant le morphisme σ par $\sigma(h) = (g+1)h$: le terme h s'identifie à la suite de terme général $n!$, puisque $(n+1)! = (n+1)n!$.

Exercice 29.3 Donner une construction analogue d'un corps à différence permettant de représenter la suite H_n de terme général $1 + \frac{1}{2} + \cdots + \frac{1}{n}$. ■

Dans la suite de cet ouvrage, nous préférons noter les éléments d'anneaux à différence à l'aide de la notation par indice utilisée pour les suites (ou encore à l'aide de la notation parenthésée pour les applications générales). Nous mentionnons soigneusement lorsqu'il ne s'agira pas de suites : ainsi, nous parlerons de $\mathbb{Q}(n, n!, \mathrm{H}_n)$ et écrirons H_{n+5} plutôt que $\sigma^5(\mathrm{H})$. Nous avons alors la définition suivante, à comparer avec la définition d'une *suite* hypergéométrique.

Définition 29.4 Soit A un anneau à différence contenant $\mathbb{Q}(n)$. Un élément t_n de A est dit *terme hypergéométrique* s'il existe une fraction rationnelle non nulle $\mathrm{R}(n)$ telle que $t_{n+1}/t_n = \mathrm{R}(n)$.

Exemple 29.2 Considérons la récurrence de la forme (29.1) obtenue pour $\mathrm{Q}(n) = n(n-1)(n-2)$ et $\mathrm{P}(n) = \mathrm{Q}(n)(n-2)$. Toutes les suites (u_n) solutions sont hypergeométriques, et de la forme

$$u_n = \begin{cases} \lambda_n & \text{si } n = 0, \dots, 2, \\ \lambda_3 (n-3)! & \text{si } n \geq 3, \end{cases}$$

pour des constantes arbitraires $\lambda_0,\dots,\lambda_3$. L'égalité $u_{n+1}/u_n = \mathrm{P}(n)/\mathrm{Q}(n) = n-2$ n'est valable en général que pour $n \geq 3$. Cependant, toutes ces suites solutions correspondent au même terme hypergéométrique h obtenu dans le corps à différence $\mathbb{Q}(g,h)$, donné par le morphisme σ satisfaisant à $\sigma(g) = g+1$ et $\sigma(h)/h = g-2$, sans notion de domaine de validité pour la formule.

Algorithme de Gosper

Étant donnée une suite hypergéométrique notée $(u(n))$, le problème de sommation indéfinie hypergéométrique de $u(n)$ consiste à déterminer s'il existe une autre suite hypergéométrique $(\mathrm{U}(n))$ telle que $\mathrm{U}(n+1)-\mathrm{U}(n) = u(n)$, et si oui, la calculer. Nous commençons par résoudre le problème analogue pour les termes hypergéométriques. Une observation simple est formulée dans le lemme suivant.

Lemme 29.1 Si $\mathrm{U}(n)$ est un terme hypergéométrique et L un opérateur de récurrence linéaire à coefficients polynomiaux, alors il existe une fraction rationnelle $r(n)$ telle que $(\mathrm{L}\cdot\mathrm{U})(n) = r(n)\mathrm{U}(n)$.

Démonstration. Si $\mathrm{U}(n)$ est hypergéométrique, par définition, il existe une fraction rationnelle $\mathrm{R}(n)$ telle que $\mathrm{U}(n+1) = \mathrm{R}(n)\mathrm{U}(n)$ et donc par récurrence $\mathrm{U}(n+k) = \mathrm{R}(n+k-1)\cdots\mathrm{R}(n)\mathrm{U}(n)$ pour tout k. Le résultat s'en déduit en additionnant les contributions. ∎

Ce lemme entraîne qu'une somme hypergéométrique $\mathrm{U}(n)$ de $u(n)$ doit être le produit de $u(n)$ par une fraction rationnelle $\mathrm{R}(n)$. En effet, dans ce cas, le Lemme 29.1 a pour conséquence l'égalité $u(n) = \mathrm{U}(n+1)-\mathrm{U}(n) = \tilde{\mathrm{R}}(n)\mathrm{U}(n)$ pour une fraction rationnelle $\tilde{\mathrm{R}}$; il suffit ensuite de considérer $\mathrm{R}(n) = 1/\tilde{\mathrm{R}}(n)$. La division terme à terme de la relation de récurrence $\mathrm{U}(n+1)-\mathrm{U}(n) = u(n)$ par $u(n)$ réduit alors le calcul à celui de la recherche de solutions rationnelles de l'équation inhomogène d'ordre 1

$$\mathrm{R}(n+1)\frac{u(n+1)}{u(n)} - \mathrm{R}(n) = 1, \tag{29.3}$$

problème traité au Chapitre 16, particulièrement en Section 16.4. Cette méthode, connue sous le nom d'algorithme de Gosper, est résumée dans l'Algorithme 29.1. Notons que puisque nous disposons d'un algorithme de décision pour la résolution rationnelle, nous obtenons un algorithme de décision aussi pour la sommation indéfinie hypergéométrique : lorsque l'algorithme renvoie $\nexists$, c'est une preuve qu'aucune somme hypergéométrique n'existe.

Exercice 29.4 Montrer qu'il y a unicité de la somme indéfinie hypergéométrique d'un terme u_n si et seulement si u_n n'est pas rationnel. ∎

Exemple 29.3 Nous voulons calculer une somme indéfinie hypergéométrique du

Entrée Un terme hypergéométrique $f(n)$.
Sortie Un terme hypergéométrique $g(n)$ tel que $g(n+1)-g(n)=f(n)$, ou le symbole $\nexists$.

1. Soit $\rho(n)$ la fraction rationnelle $f(n+1)/f(n)$.
2. Résoudre la récurrence $\rho(n)\mathrm{R}(n+1)-\mathrm{R}(n)=1$ par l'Algorithme 16.7.
3. Si la résolution fournit une solution, renvoyer $\mathrm{R}(n)f(n)$, sinon renvoyer $\nexists$.

Algorithme 29.1 – Algorithme de Gosper pour la sommation hypergéométrique indéfinie.

terme $\dfrac{2^n(n-1)}{n(n+1)}$. Pour ceci, nous résolvons (29.3), qui prend ici la forme

$$2n^2\mathrm{R}(n+1)-(n-1)(n+2)\mathrm{R}(n)=(n-1)(n+2),$$

et recherchons ses solutions rationnelles. Nous utilisons l'algorithme d'Abramov, en commençant par l'Algorithme 16.6 pour une borne sur les dénominateurs possibles. Ici, le coefficient de tête décalé vers l'arrière de l'ordre 1, $(n-1)^2$, ne peut avoir de pgcd commun avec le coefficient de queue décalé vers l'avant de $h\in\mathbb{N}$, $(n+h-1)(n+h+2)$, que pour $h=0$. Nous avons donc, sans calcul de résultant, que l'Algorithme 16.6 trouve $m=1$, $h_1=0$, et renvoie $\mathrm{Q}(n)=n-1$. Écrivons $\mathrm{R}(n)=\mathrm{P}(n)/(n-1)$ pour un polynôme P à déterminer. Par changement de fonction inconnue, la récurrence devient

$$2n\mathrm{P}(n+1)-(n+2)\mathrm{P}(n)=(n-1)(n+2).$$

Comme $n+2$ divise deux des termes polynomiaux, il divise le troisième, si bien que P s'écrit sous la forme $\mathrm{P}(n)=(n+1)\mathrm{Q}(n)$ pour un polynôme Q à déterminer. Par un second changement de fonction inconnue, la récurrence devient

$$2n\mathrm{Q}(n+1)-(n+1)\mathrm{Q}(n)=n-1.$$

Si $\mathrm{Q}(n)\sim\lambda n^\alpha$ à l'infini, alors $\lambda n^{\alpha+1}\sim n$, et donc $\alpha=0$, $\mathrm{Q}(n)=\lambda=1$. Nous avons donc finalement trouvé $\mathrm{R}(n)$ sous la forme $(n+1)/(n-1)$, ce qui fournit la somme indéfinie hypergéométrique $2^n/n$.

Lien avec la sommation des suites

L'Algorithme 29.3, que nous allons décrire, donne la somme indéfinie hypergéométrique, quand elle existe, d'un *terme* hypergéométrique. Tout d'abord nous faisons le lien avec le problème initial sur les *suites* hypergéométriques, dont les *termes* hypergéométriques ne sont qu'une idéalisation. En particulier, nous allons rectifier l'image selon laquelle une somme hypergéométrique se réduit à un multiple rationnel de la suite sommée.

Pour une suite $(u_n)_{n\in\mathbb{N}}$ solution de (29.1), introduisons la fraction rationnelle $\rho = P/Q \in \mathbb{Q}(n)$, dont les pôles sont des zéros du polynôme $Q(n)$. Soit R la fraction rationnelle calculée dans l'Algorithme 29.1 de Gosper. En réinterprétant les fractions rationnelles comme des fonctions, nous avons :

— $u_{n+1} = \rho(n)u_n$ si $Q(n) \neq 0$,
— $\rho(n)R(n+1) - R(n) = 1$ si n est dans l'intersection des domaines de définition de $\rho(n)$, $R(n)$ et $R(n+1)$.

Définissons une suite $\tilde{U}$ par

$$\tilde{U}_n = \begin{cases} R(n)u_n & \text{si } n \text{ est dans le domaine de définition de R,} \\ 0 & \text{sinon.} \end{cases}$$

Nous avons alors :

$$\tilde{U}_{n+1} - \tilde{U}_n = R(n+1)u_{n+1} - R(n)u_n = \big(\rho(n)R(n+1) - R(n)\big)u_n = u_n,$$

où la première égalité est vérifiée si n et $n+1$ sont dans le domaine de définition de R, et où les suivantes le sont si de plus $Q(n) \neq 0$. Soit S l'ensemble (fini) des zéros de $Q(n)$ et des pôles de $R(n)$ et de $R(n+1)$ dans $\mathbb{N}$, si bien qu'en résumé :

$$\tilde{U}_{n+1} - \tilde{U}_n = u_n \quad \text{si } n \in \mathbb{N} \setminus S.$$

Définissons deux suites (ϕ_n) et (Φ_n) par

$$\phi_n = \begin{cases} u_n - (\tilde{U}_{n+1} - \tilde{U}_n) & \text{si } n \in S, \\ 0 & \text{sinon,} \end{cases}$$

et $\Phi_n = \sum_{i=0}^{n-1} \phi_i$. Cette dernière suite est constante par morceaux et la partition de $\mathbb{N}$ servant à sa définition est finie et calculable. Posons finalement

$$U_n = \tilde{U}_n + \Phi_n,$$

de sorte que pour tout $n \in \mathbb{N}$, la relation $U_{n+1} - U_n = \tilde{U}_{n+1} - \tilde{U}_n + \phi_n = u_n$ est vérifiée : cette suite (U_n) est une somme indéfinie sur tout $\mathbb{N}$ de (u_n). Le résultat qui suit résume cette discussion.

Proposition 29.2 Si une suite hypergéométrique $(u_n)_{n\in\mathbb{N}}$ admet une somme indéfinie hypergéométrique $(U_n)_{n\in\mathbb{N}}$ sur tout $\mathbb{N}$, cette dernière s'écrit sous la forme

$$U_n = [\![R(n)u_n]\!] + \Phi_n$$

où :
— la notation $[\![\dots]\!]$ est utilisée pour étendre en une suite totale sur $\mathbb{N}$ une suite partiellement définie, en la prolongeant par 0,
— la suite $(\Phi_n)_{n\in\mathbb{N}}$ est une suite constante par morceaux définie sur une partition finie de $\mathbb{N}$ obtenue à l'aide des pôles de R et zéros de Q.

Exemple 29.4 L'Algorithme 29.1 de Gosper permet également de donner des réponses négatives. Voici en détail comment il permet de prouver par l'absurde que la suite des $S(n) := \sum_{k=1}^{n} 1/k!$ n'est pas hypergéométrique.

Supposons que $S(n)$ est hypergéométrique. Alors, elle doit être le produit de $1/n!$ par une fraction rationnelle $r(n)$. Cette fraction est donc solution de la récurrence

$$S(n+1) - S(n) = \frac{1}{(n+1)!} = \frac{r(n+1)}{(n+1)!} - \frac{r(n)}{n!}.$$

En chassant les dénominateurs, il reste $r(n+1) - (n+1)r(n) = 1$. Le résultat de l'Algorithme 16.6 d'Abramov page 304 montre que r doit être un polynôme : les pôles de plus petite partie réelle de r doivent être des zéros du pgcd, trivial, de 1 et $n+1$. Enfin, r ne peut pas non plus être un polynôme : son terme de plus haut degré ne disparaît pas par évaluation de la récurrence.

Sommation paramétrée

Soient des termes $u_1(n), \ldots, u_s(n)$ hypergéométriques et deux à deux similaires, au sens où tous les rapports $u_i(n)/u_j(n)$ sont rationnels. Il s'agit de déterminer, s'ils existent, un terme hypergéométrique $S(n)$ et des constantes $\lambda_1, \ldots, \lambda_s$ telles que

$$S(n+1) - S(n) = \lambda_1 u_1(n) + \cdots + \lambda_s u_s(n).$$

Comme toutes les u_i sont similaires à une même suite $u(n)$, le membre droit de (29.3) peut être remplacé par une combinaison linéaire des λ_i à coefficients des fractions rationnelles et la méthode de la Section 16.4 page 306 pour les récurrences linéaires inhomogènes paramétrées s'applique.

29.2 Sommation hypergéométrique définie. Algorithme de Zeilberger

L'algorithme considéré dans cette section s'applique à des suites $u(n,k)$ dites hypergéométriques en deux variables, c'est-à-dire pour lesquelles

$$\frac{u(n+1,k)}{u(n,k)} \quad \text{et} \quad \frac{u(n,k+1)}{u(n,k)}$$

sont deux fractions rationnelles en n et k, au moins en dehors du lieu d'annulation d'un polynôme non nul fixé $Z(n,k)$.

Exercice 29.5 Vérifier que la suite $u(n,k) = \binom{n}{k}$ est hypergéométrique en ce sens. ■

Exercice 29.6 Soit une suite hypergéométrique $(u_{n,k})$, donnée par les rapports rationnels $a(n,k) = u_{n+1,k}/u_{n,k}$ et $b(n,k) = u_{n,k+1}/u_{n,k}$. Montrer que les fractions rationnelles a et b vérifient une relation que l'on explicitera. ■

Le problème de sommation définie hypergéométrique est de déterminer si pour une telle suite $(u(n))$, la nouvelle suite de terme général

$$U(n) = \sum_k u(n,k)$$

vérifie une récurrence linéaire et si oui, la calculer. L'absence de bornes explicites de sommation signifie que la somme porte en fait sur toutes les valeurs $k \in \mathbb{Z}$, étant entendu que, bien souvent, ces sommes ont en fait un support fini. Bien souvent aussi, l'intervalle de sommation s'arrête aux *bornes naturelles* de sommation, c'est-à-dire aux indices où par construction la suite et toutes ses décalées par rapport à l'autre indice, n, s'annulent.

Exemple 29.5 Des sommes simples sont

$$\sum_k \binom{n}{k} = 2^n \quad \text{et} \quad \sum_k \binom{n}{k}^2 = \binom{2n}{n}.$$

Dans ces deux exemples, pour k en dehors de $\{0,\dots,n\}$, les binomiaux sont nuls. Ainsi, les bornes 0 et n sont naturelles.

Un cas très favorable pour obtenir des sommations explicites est celui où $\mathrm{U}(n)$ est elle-même hypergéométrique. Mais pour bien des applications, il est tout aussi utile de trouver une récurrence linéaire à laquelle $\mathrm{U}(n)$ obéit. C'est ce que réalise l'algorithme de Zeilberger, par une approche dite de *télescopage créatif*. Cette approche ne garantit cependant pas d'obtenir une relation de récurrence d'ordre minimal pour la somme ; elle ne détecte donc pas directement si la somme est elle aussi hypergéométrique. Un test de cette dernière situation se ramène à la recherche de solutions hypergéométriques de récurrences linéaires, problème qui est traité par l'algorithme de Petkovšek en Section 29.3.

Principe du télescopage créatif

Désignons par S_n et S_k les opérateurs de décalage en n et k, de sorte que $(\mathrm{S}_n \cdot u)(n,k) = u(n+1,k)$ et $(\mathrm{S}_k \cdot u)(n,k) = u(n,k+1)$. On cherche un opérateur de la forme

$$\mathrm{P}(n,\mathrm{S}_n) - (\mathrm{S}_k - 1)\mathrm{Q}(n,k,\mathrm{S}_n,\mathrm{S}_k), \tag{29.4}$$

polynomial en S_n et S_k, et qui annule la suite $(u(n,k))$ à sommer. Une fois un tel opérateur trouvé, la sommation sur k est aisée, et, sous l'hypothèse simplificatrice de bornes naturelles, conduit à l'égalité

$$\big(\mathrm{P}(n,\mathrm{S}_n)\cdot \mathrm{U}\big)(n) = 0. \tag{29.5}$$

Exemple 29.6 En suivant ce modèle, voici une preuve compliquée que

$$\mathrm{U}(n) = \sum_k \binom{n}{k} = 2^n. \tag{29.6}$$

Le point de départ est l'identité du triangle de Pascal

$$\binom{n+1}{k+1} = \binom{n}{k+1} + \binom{n}{k},$$

qui s'interprète en l'opérateur annulateur

$$\mathrm{S}_n\mathrm{S}_k - \mathrm{S}_k - 1. \tag{29.7}$$

Ce dernier se récrit

$$(\mathrm{S}_n - 2) + (\mathrm{S}_k - 1)(\mathrm{S}_n - 1),$$

opérateur qui traduit la relation explicite

$$\left(\binom{n+1}{k} - 2\binom{n}{k}\right) + \left(\binom{n+1}{k+1} - \binom{n+1}{k}\right) - \left(\binom{n}{k+1} - \binom{n}{k}\right) = 0.$$

En sommant sur $k \in \mathbb{Z}$, les premiers termes se téléscopent deux à deux, et il reste

$$((\mathrm{S}_n - 2)\cdot \mathrm{U})(n) = \mathrm{U}(n+1) - 2\mathrm{U}(n) = 0.$$

La fin de la preuve se ramène à vérifier la condition initiale $\mathrm{S}(0) = 1$.

Notons que les coefficients de l'opérateur (29.7) obtenu pour cet exemple sont constants, mais que dans le cas de sommations plus générales qui nous occupent dans ce chapitre, ils pourront être des fractions rationnelles en n et k.

En l'absence de bornes naturelles, l'équation (29.5) s'écrit plutôt

$$(\mathrm{P}\cdot \mathrm{U})(n) = (\mathrm{Q}\cdot u)(n, b+1) - (\mathrm{Q}\cdot u)(n, a).$$

Les opérations de clôture vues au Chapitre 14 fournissent alors un opérateur $\tilde{\mathrm{P}}(n, \mathrm{S}_n)$ qui annule le membre droit de l'égalité ci-dessus. On obtient ainsi une équation homogène, sous la forme

$$(\tilde{\mathrm{P}}\mathrm{P})\cdot \mathrm{U} = 0.$$

Algorithme de Zeilberger

Il reste à voir comment trouver les opérateurs P et Q de l'équation (29.4) dans le cas général. L'idée de Zeilberger est que cette équation,

$$\big(\mathrm{P}(n, \mathrm{S}_n)\cdot u\big)(n,k) = \big((\mathrm{S}_k - 1)\mathrm{Q}(n,k,\mathrm{S}_n,\mathrm{S}_k)\cdot u\big)(n,k),$$

signifie que $\mathrm{Q}\cdot u$ est une somme hypergéométrique *indéfinie* de $\mathrm{P}(n,\mathrm{S}_n)\cdot u$ par rapport à k. Comme u est supposée hypergéométrique, $\mathrm{Q}\cdot u$ est en fait de la forme $q(n,k)u(n,k)$ pour une fraction rationnelle q : si P était connu, l'Algorithme 29.1 de Gosper suffirait pour trouver q.

La méthode conçue par Zeilberger procède alors incrémentalement sur le degré de P en S_n. Pour $r = 0, 1, \ldots$, elle cherche s'il existe des coefficients $\lambda_i(n)$ pour P tels que

$$\lambda_0(n)u(n,k) + \lambda_1(n)u(n+1,k) + \cdots + \lambda_r(n)u(n+r,k)$$

Entrée Un terme hypergéométrique $u(n,k)$ et un entier $m \in \mathbb{N}$.
Sortie Une famille $(\lambda_i(n))_{0\le i\le r}$ de fractions rationnelles, avec $r \le m$, et une fraction rationnelle $\mathrm{R}(n,k)$ telles que

$$\lambda_r(n)u_{n+r,k} + \cdots + \lambda_0(n)u_{n,k} = \mathrm{R}(n,k+1)u_{n,k+1} - \mathrm{R}(n,k)u_{n,k},$$

ou le symbole $\nexists$.

1. Pour r de 0 à m :
   a. fixer n comme un paramètre et introduire les termes hypergéométriques $u_i(k) = u(n+i,k)$, pour $0 \le i \le r$;
   b. par l'algorithme de la Section 29.1, résoudre

$$\mathrm{S}(k+1) - \mathrm{S}(k) = \lambda_0 u_0(k) + \cdots + \lambda_r u_r(k)$$

   en un terme hypergéométrique $\mathrm{S}(k) = \mathrm{R}(n,k)u(k)$ pour une fraction rationnelle R et en des fractions rationnelles $\lambda_i = \lambda_i(n)$ indépendantes de k ;
   c. si la résolution fournit une solution, renvoyer la famille $(\lambda_i)_{i=0}^{r}$ et $\mathrm{R}(n,k)$.
2. Renvoyer $\nexists$.

Algorithme 29.2 – Algorithme de Zeilberger pour la sommation hypergéométrique définie.

ait une somme hypergéométrique. L'algorithme pour cela est la variante paramétrée de l'algorithme de Gosper, présentée en Section 29.1.

Exercice 29.7 Exécuter l'algorithme de Zeilberger sur la somme (29.6) de l'Exemple 29.6. Comparer les résultats des calculs et les preuves de l'identité selon l'approche utilisant la relation du triangle de Pascal et celle utilisant l'algorithme de Zeilberger. ■

Exercice 29.8 Évaluer la somme hypergéométrique

$${}_2\mathrm{F}_1\left(\begin{matrix}-n, b\\ c\end{matrix}\,\middle|\, 1\right)$$

en termes du symbole de Pochhammer $(x)_n$ donné par la Définition 29.2. ■

Exercice 29.9 On veut montrer, pour tout $n \in \mathbb{N}$, l'identité

$${}_2\mathrm{F}_1\left(\begin{matrix}-2n, 2n+2\alpha+1\\ \alpha+1\end{matrix}\,\middle|\, \frac{1-x}{2}\right) = {}_2\mathrm{F}_1\left(\begin{matrix}-n, n+\alpha+\frac{1}{2}\\ \alpha+1\end{matrix}\,\middle|\, 1-x^2\right).$$

Indiquer (sans faire les calculs) comment on peut utiliser l'algorithme de Zeilberger pour arriver à ce résultat. ■

Terminaison

La méthode incrémentale appliquée avec un degré r de P non borné ne termine pas en général, d'où le paramètre m en entrée de l'Algorithme 29.2. Wilf et Zeilberger ont été les premiers à délimiter une classe importante, celle des suites (respectivement termes) *proprement hypergéométriques* sur lesquelles la terminaison et par conséquent le succès de la méthode sont garantis. Ces suites (respectivement termes) sont de la forme

$$u(n,k) = \mathrm{P}(n,k)\mathrm{Z}^k \prod_{\ell=1}^{\mathrm{L}} (a_\ell n + b_\ell k + c_\ell)!^{e_\ell}, \tag{29.8}$$

où les a_ℓ, b_ℓ et e_ℓ sont des *entiers*, L est un entier positif, P un polynôme et Z une constante.

Proposition 29.3 Une variante sans borne m en entrée de l'Algorithme 29.2 de Zeilberger termine si l'entrée $u(n,k)$ est proprement hypergéométrique.

Démonstration. La preuve repose sur un résultat légèrement plus fort, à savoir l'existence d'un polynôme non nul $\mathrm{A}(n,\mathrm{S}_k,\mathrm{S}_n)$ qui ne dépend pas de k et annule $u(n,k)$; l'existence d'un tel polynôme A est justifiée par le Lemme 29.4 ci-après. Par la suite, des divisions euclidiennes successives de l'opérateur A par $\mathrm{S}_k - 1$, sous la forme $\mathrm{A} = (\mathrm{S}_k - 1)^p[(\mathrm{S}_k - 1)\tilde{\mathrm{Q}}(n,\mathrm{S}_k,\mathrm{S}_n) + \tilde{\mathrm{P}}(n,\mathrm{S}_n)]$ avec $\tilde{\mathrm{P}} \neq 0$, vont fournir un couple (P,Q) avec $\mathrm{P} \neq 0$ tel que l'opérateur (29.4) annule $u(n,k)$ et l'opérateur Q ne dépend pas de k. En effet, en appliquant A à $u(n,k)$ et en multipliant par $k(k-1)\cdots(k-p+1)$ à gauche, on obtient

$$k(k-1)\cdots(k-p+1)\mathrm{A}\cdot u(n,k)$$

qui vaut

$$(-1)^p p!\tilde{\mathrm{P}}\cdot u(n,k) + (\mathrm{S}_k - 1)\cdot v(n,k)$$

pour un nouveau terme $v(n,k)$. Comme le P obtenu, c'est-à-dire $(-1)^p p!\tilde{\mathrm{P}}$, est non nul, la recherche exhaustive par degré croissant en S_n, telle que mise en place par l'Algorithme 29.2, doit terminer. ∎

Lemme 29.4 Pour un terme proprement hypergéométrique $u(n,k)$ de la forme (29.8), soient

$$\alpha = \sum_{\ell=1}^{\mathrm{L}} |e_\ell a_\ell| \qquad \text{et} \qquad \beta = \sum_{\ell=1}^{\mathrm{L}} |e_\ell b_\ell|.$$

Il existe un opérateur annulateur $\mathrm{A}(n,\mathrm{S}_n,\mathrm{S}_k)$ non nul d'ordres de décalage relativement à n et k respectivement bornés par

$$r = 2\beta \qquad \text{et} \qquad s = 2(2\alpha-1)\beta + \deg_k \mathrm{P} + 1. \tag{29.9}$$

Démonstration. Pour $0 \le i \le r$ et $0 \le j \le s$, le décalage $u(n+i,k+j)$ fait intervenir des factorielles de la forme $(a_\ell n + b_\ell k + c_\ell + \delta)!$ pour un décalage δ borné en valeur absolue par $\sigma_\ell = |a_\ell| r + |b_\ell| s$. Pour $e_\ell > 0$, observons que $(a_\ell n + b_\ell k + c_\ell + \delta)!/(a_\ell n + b_\ell k + c_\ell - \sigma_\ell)!$

est un polynôme en k de degré au plus $2\sigma_\ell$; pour $e_\ell < 0$, observons qu'il en est de même de $(a_\ell n + b_\ell k + c_\ell + \sigma_\ell)!/(a_\ell n + b_\ell k + c_\ell + \delta)!$. Ainsi, pour les $(r+1)(s+1)$ choix de (i,j),

$$\frac{\prod_{1\le\ell\le L, e_\ell<0}(a_\ell n + b_\ell k + c_\ell + \sigma_\ell)!}{\prod_{1\le\ell\le L, e_\ell>0}(a_\ell n + b_\ell k + c_\ell - \sigma_\ell)!}\,\frac{u(n+i,k+j)}{u(n,k)}$$

est un polynôme en k de degré au plus $\deg_k P + 2\sum_{\ell=1}^{L}|e_\ell|\sigma_\ell \le \deg_k P + 2(\alpha r + \beta s)$. Il s'ensuit qu'il existe pour r et s assez grand une relation de dépendance linéaire entre les $u(n+i,k+j)$, à coefficients dans $\mathbb{C}(n)$. C'est plus précisément le cas dès que $(r+1)(s+1) > \deg_k P + 2(\alpha r + \beta s) + 1$, qui est réalisé dès qu'on a (29.9). La relation de dépendance linéaire fournit A. ■

29.3 Solutions hypergéométriques. Algorithme de Petkovšek

Si $(L\cdot u)(n) = 0$ avec $u(n)$ hypergéométrique, il existe deux polynômes P et Q vérifiant

$$u(n+1) = \frac{P(n)}{Q(n)}u(n),$$

et on peut prendre Q unitaire. Il s'ensuit que

$$u(n+\ell) = \frac{P(n+\ell-1)}{Q(n+\ell-1)}\cdots\frac{P(n)}{Q(n)}u(n).$$

Pour une récurrence donnée par un opérateur $L = a_r S_n^r + \cdots + a_0$, une condition nécessaire et suffisante d'existence de solution hypergéométrique s'écrit

$$\begin{aligned} a_r(n)P(n+r-1)P(n+r-2)\cdots P(n) + \\ a_{r-1}(n)Q(n+r-1)P(n+r-2)\cdots P(n) + \cdots \\ a_1(n)Q(n+r-1)\cdots Q(n-1)P(n) + \\ a_0(n)Q(n+r-1)\cdots Q(n) = 0. \end{aligned} \tag{29.11}$$

Si on savait prédire que $\operatorname{pgcd}\big(P(n),Q(n+i)\big) = 1$ pour $i = 0,\dots,r-1$, alors on en déduirait facilement que $P(n)$ divise $a_0(n)$ et $Q(n+r-1)$ divise $a_r(n)$. Ceci nous donnerait un algorithme : toute paire de facteurs unitaires de $a_0(n)$ et $a_r(n-r+1)$ donnerait un candidat pour $Q(n)$ et un candidat pour $P(n)$ à une constante près ; il suffirait alors d'injecter ces candidats dans (29.11) et de chercher s'il existe une constante satisfaisant à l'équation. En itérant sur les facteurs de a_0 et a_r, le travail serait terminé.

En général, il se peut très bien que $P(n)$ et $Q(n+i)$ aient des facteurs communs. La solution trouvée par Petkovšek consiste à recourir à une décomposition plus forte de la fraction rationnelle. On cherche une solution telle que

$$\frac{u(n+1)}{u(n)} = Z\frac{A(n)}{B(n)}\frac{C(n+1)}{C(n)},$$

où Z est une constante, A, B, C sont des polynômes unitaires, $\operatorname{pgcd}(A(n),C(n)) = 1$, $\operatorname{pgcd}\big(B(n),C(n+1)\big) = 1$ et $\operatorname{pgcd}\big(A(n),B(n+i)\big) = 1$ pour tout $i\in\mathbb{N}$. Cette décomposition des fractions rationnelles est appelée décomposition de Gosper–Petkovšek.

Entrée Une équation de récurrence $a_r(n)u(n+r)+\cdots+a_0(n)u(n)=0$ à coefficients $a_i(n)$ polynomiaux.
Sortie L'ensemble des termes hypergéométriques t_n solutions de la récurrence.

1. Initialiser S avec l'ensemble vide.
2. Factoriser les polynômes a_0 et a_r.
3. Pour chaque polynôme $\mathrm{A}(n)$ divisant $a_0(n)$ et pour chaque polynôme $\mathrm{B}(n)$ divisant $a_r(n-r+1)$:
   a. extraire le coefficient dominant $d(\mathrm{Z})$ de
   $$\sum_{i=0}^{r} \mathrm{Z}^i a_i(n)\mathrm{B}(n+r-1)\cdots\mathrm{B}(n+i+1)\mathrm{A}(n+i)\cdots\mathrm{A}(n)\mathrm{C}(n+i)\,; \tag{29.10}$$
   b. factoriser $d(\mathrm{Z})$;
   c. pour chaque facteur irréductible $p(\mathrm{Z})$ de $d(\mathrm{Z})$:
      i. fixer ζ vérifiant $p(\zeta)=0$ en étendant au besoin le corps de constantes,
      ii. calculer les polynômes $\mathrm{C}(n)$ annulant la récurrence (29.10) par l'Algorithme 16.5 de résolution page 302.
      iii. pour chaque solution C obtenue, définir un terme hypergéométrique t_n par
      $$t_{n+1}/t_n = \zeta\frac{\mathrm{A}(n)}{\mathrm{B}(n)}\frac{\mathrm{C}(n+1)}{\mathrm{C}(n)}$$
      et ajouter ces termes à S ;
4. Renvoyer S.

Algorithme 29.3 – Algorithme de Petkovšek pour la recherche des solutions hypergéométriques d'une équation de récurrence linéaire.

Exercice 29.10 Montrer que toute fraction rationnelle peut se décomposer sous la forme ci-dessus. Donner un algorithme calculant les polynômes A, B, C et les constantes Z correspondant à une fraction rationnelle donnée en entrée. On pourra représenter Z par son polynôme annulateur minimal. (Indice : s'inspirer de l'Algorithme 16.6 d'Abramov.) ■

Avec cette décomposition, l'équation (29.11) devient

$$\begin{aligned}\mathrm{Z}^r a_r(n)\mathrm{A}(n+r-1)\cdots\mathrm{A}(n)\mathrm{C}(n+r)+\\ \mathrm{Z}^{r-1}a_{r-1}(n)\mathrm{B}(n+r-1)\mathrm{A}(n+r-2)\cdots\mathrm{A}(n)\mathrm{C}(n+r-1)+\cdots+\\ \mathrm{Z}a_1(n)\mathrm{B}(n+r-1)\cdots\mathrm{B}(n-1)\mathrm{A}(n)\mathrm{C}(n+1)+\\ a_0(n)\mathrm{B}(n+r-1)\cdots\mathrm{B}(n)\mathrm{C}(n)=0.\end{aligned} \tag{29.12}$$

Les contraintes de la décomposition permettent de déduire immédiatement

$$\mathrm{A}(n) \mid a_0(n), \qquad \mathrm{B}(n+r-1) \mid a_r(n).$$

L'algorithme de Petkovšek s'en déduit : pour chaque paire (A, B) de facteurs de a_0 et a_r, le coefficient de tête du polynôme au membre gauche de (29.12) donne une équation polynomiale sur Z. Une fois Z ainsi fixé, il reste à chercher les solutions polynomiales C de l'équation, s'il en existe. La méthode est résumée dans l'Algorithme 29.3.

Exercice 29.11 Utiliser l'Algorithme 29.3 pour résoudre

$$(n-1)u(n+2) - (n^2+3n-2)u(n+1) + 2n(n+1)u(n) = 0,$$
$$u(n+2) - (2n+1)u(n+1) + (n^2-2)u(n) = 0.$$

■

On peut noter que la recherche de solutions hypergéométriques de récurrences linéaires *inhomogènes* n'est pas difficile : le membre droit doit être hypergéométrique et, outre les solutions hypergéométriques de la partie homogène, la solution doit être le produit du membre droit par une fraction rationnelle. Ceci ramène le problème à la recherche de solutions rationnelles.

Notes

Le livre de Petkovšek, Wilf et Zeilberger intitulé A = B [PWZ96] est une bonne introduction aux questions abordées dans ce chapitre. L'algorithme de Petkovšek possède une extension intéressante [AP94], par réduction de l'ordre, à une plus grande classe de solutions, appelées *solutions d'Alembertiennes*.

La présentation traditionnelle [Gos78] de l'Algorithme 29.1 de Gosper est un peu différente en ce qu'elle ne fait pas appel à un algorithme général de résolution rationnelle : la description usuelle développe directement la prédiction d'un facteur du numérateur (voir la fin de la Section 16.4).

Le traitement de la Section 29.1 s'inspire de la présentation élémentaire par Kauers et Paule [KP11, Section 3.5.4] ; il a sa source dans des travaux d'Abramov et Petkovšek [AP05], qui montrent de plus que la somme indéfinie U_n pourra être prise directement comme le produit $\mathrm{R}(n)u_n$, sans apparition du terme Φ_n de la Proposition 29.2 et juste après un éventuel prolongement par continuité, à chaque fois que la suite (u_n) sera donnée comme évaluation aux entiers d'une fonction de la variable complexe analytique au voisinage de $\mathbb{N}$ [AP07].

La Proposition 29.3 ne donne qu'une condition suffisante pour l'existence d'un $\mathrm{P}(n, \mathrm{S}_n)$ et conséquemment pour la terminaison de l'algorithme de Zeilberger. Il existe des suites qui ne sont pas proprement hypergéométriques et pour lesquelles la méthode fonctionne quand même. Cela a donné lieu à une série de travaux, et le dernier mot revient à un travail d'Abramov [Abr03] qui donne un test algorithmique de terminaison de l'algorithme.

De nombreuses généralisations de l'algorithme de Zeilberger sont possibles : par exemple la suite à sommer peut ne pas être hypergéométrique, tout en étant polynomialement récurrente, ou il peut s'agir d'une suite de fonctions différentiellement

finies et l'on cherche une équation différentielle satisfaite par la somme définie, etc. Ces questions seront abordées au Chapitre 32.

L'algorithme de Petkovšek permet aussi de décrire l'espace des solutions d'une équation de récurrence linéaire qui sont combinaisons linéaires de termes hypergéométriques sans être elles-mêmes réduites à de simples termes hypergéométriques [Pet92].

Bibliographie

Abr03 Abramov, S. A. (2003). « When does Zeilberger's algorithm succeed ? » In : *Advances in Applied Mathematics*, vol. 30, n°3, p. 424–441.

AP05 Abramov, S. A. et M. Petkovšek (2005). « Gosper's algorithm, accurate summation, and the discrete Newton–Leibniz formula ». In : *ISSAC'05 : International Symposium on Symbolic and Algebraic Computation*. ACM, New York, p. 5–12.

AP07 Abramov, Sergei A. et Marko Petkovšek (2007). « Analytic solutions of linear difference equations, formal series, and bottom summation ». In : *Computer Algebra in Scientific Computing, 10th International Workshop, CASC 2007, Bonn, Germany, September 16-20, 2007, Proceedings*, p. 1–10.

AP94 — (1994). « D'Alembertian solutions of linear differential and difference equations ». In : *ISSAC'94 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 169–174.

Gos78 Gosper, R. William (1978). « Decision procedure for indefinite hypergeometric summation ». In : *Proceedings of the National Academy of Sciences USA*, vol. 75, n°1, p. 40–42.

KP11 Kauers, Manuel et Peter Paule (2011). *The concrete tetrahedron*. Texts and Monographs in Symbolic Computation. Symbolic sums, recurrence equations, generating functions, asymptotic estimates. Springer, Vienna.

Pet92 Petkovšek, Marko (1992). « Hypergeometric solutions of linear recurrences with polynomial coefficients ». In : *Journal of Symbolic Computation*, vol. 14, n°2-3, p. 243–264.

PWZ96 Petkovšek, Marko, Herbert S. Wilf et Doron Zeilberger (1996). A = B. A. K. Peters.

30. Équations fonctionnelles linéaires et polynômes tordus

Résumé

Une certaine variété de polynômes non commutatifs fournit une représentation unifiée pour une large classe d'équations fonctionnelles linéaires. Celle-ci s'avère bien adaptée pour les calculs. Nous réinterprétons de ce point de vue nombre des algorithmes vus dans les chapitres précédents.

30.1 Polynômes non commutatifs pour calculer avec des opérateurs linéaires

Dans cette section, nous présentons un cadre unifié pour la manipulation de systèmes linéaires liant des dérivées $f_i^{(j)}(x)$, des décalées $f_i(x+j)$, ou les substitutions $f_i(q^j x)$ de fonctions inconnues $f_i(x)$. À cette fin, nous introduisons de nouvelles familles de polynômes en une indéterminée ayant la propriété que cette indéterminée ne commute pas avec les coefficients des polynômes. Ce défaut de commutativité reflète une sorte de loi de Leibniz.

Rappelons la relation de Leibniz pour deux fonctions quelconques f et g :

$$(fg)'(x) = f'(x)g(x) + f(x)g'(x).$$

En notant D l'opérateur de dérivation, M celui qui à une fonction f associe la fonction donnée par $\mathrm{M}(f)(x) = xf(x)$, id l'opérateur identité sur les fonctions, et $\circ$ la composition d'opérateurs, la règle de Leibniz donne, pour $f(x) = x$ et g quelconque,

$$(\mathrm{D}\circ\mathrm{M})(g) = \mathrm{D}\big(\mathrm{M}(g)\big) = \mathrm{M}\big(\mathrm{D}(g)\big) + g = (\mathrm{M}\circ\mathrm{D} + \mathrm{id})(g).$$

L'identité étant vérifiée par toute fonction g, on obtient l'égalité $\mathrm{D}\circ\mathrm{M} = \mathrm{M}\circ\mathrm{D} + \mathrm{id}$ entre opérateurs linéaires différentiels. D'autres opérateurs vérifient des analogues de

la règle de Leibniz : l'opérateur Δ de différence finie, défini par $\Delta(f)(x) = f(x+1) - f(x)$; l'opérateur $S = \Delta + \mathrm{id}$ de décalage, défini par $S(f)(x) = f(x+1)$; pour une constante q fixée autre que 0 et 1, l'opérateur H de dilatation, défini par $H(f)(x) = f(qx)$. On a les relations :

$$\begin{aligned} \Delta(fg)(x) &= f(x+1)\Delta(g)(x) + \Delta(f)(x)g(x), \\ (fg)(x+1) &= f(x+1)g(x+1), \\ (fg)(qx) &= f(qx)g(qx), \end{aligned}$$

qui mènent aux relations $\Delta \circ M = (M+\mathrm{id}) \circ \Delta + \mathrm{id}$, $S \circ M = (M+\mathrm{id}) \circ S$, $H \circ M = Q \circ M \circ H$ entre opérateurs linéaires, après avoir introduit un nouvel opérateur Q donné par $Q(f)(x) = qf(x)$. La définition suivante fournit un même cadre algébrique pour ces différents contextes d'opérateurs.

Définition 30.1 Soit A un anneau commutatif intègre de caractéristique zéro, que nous supposons muni d'un endomorphisme injectif σ et d'une σ-dérivation δ, au sens où pour tout a et tout b de A,

$$\sigma(a+b) = \sigma(a) + \sigma(b), \qquad \sigma(ab) = \sigma(a)\sigma(b), \qquad \delta(ab) = \sigma(a)\delta(b) + \delta(a)b.$$

Pour une nouvelle indéterminée ∂, on appelle *anneau de polynômes tordus* l'algèbre sur A engendrée par ∂ et les relations, pour tout a de A,

$$\partial a = \sigma(a)\partial + \delta(a).$$

On note cet anneau $A\langle \partial; \sigma, \delta \rangle$.

Une conséquence immédiate de la définition est que tout élément f d'un anneau de polynômes tordus admet une récriture canonique de la forme

$$f = a_r \partial^r + \cdots + a_0$$

pour des a_i dans A, avec a_r non nul sauf si la somme ne comprend aucun terme (et alors $f = 0$). Les a_i et l'entier r ainsi définis (sauf pour $f = 0$) sont uniques et r a les propriétés d'un degré. En particulier, le degré d'un produit est la somme des degrés de ses facteurs. Ceci résulte de l'injectivité de σ et de l'intégrité, par la formule

$$(a\partial^r + \cdots)(b\partial^s + \cdots) = a\sigma^r(b)\partial^{r+s} + \cdots$$

car $a\sigma^r(b)$ est nul si et seulement si a ou b l'est.

Des choix adéquats de σ et δ nous font retrouver les quelques exemples donnés plus haut. Pour simplifier la notation, nous supposons que A peut s'identifier à un bon espace de fonctions. En notant 0 l'application qui a toute fonction associe la fonction constante nulle, on trouve en particulier les exemples de la Figure 30.1. Toutes les algèbres d'opérateurs de cette figure sont à coefficients dans $\mathbb{Q}(x)$; on dispose aussi d'analogues pour $A = \mathbb{Q}[x]$ et, quand $\sigma \neq S$, pour $A = \mathbb{Q}[x, x^{-1}]$.

On fera attention à la notation. Si la composition entre opérateurs est notée par $\circ$, nous ne ferons qu'une simple juxtaposition pour le produit de polynômes tordus, et

Anneau de polynômes tordus	Natures d'opérateurs représentés
$\mathbb{Q}(x)\langle\partial;\mathrm{id},\mathrm{D}\rangle$	opérateurs différentiels linéaires
$\mathbb{Q}(x)\langle\partial;\mathrm{S},0\rangle$	opérateurs de récurrence
$\mathbb{Q}(x)\langle\partial;\mathrm{S},\Delta\rangle$	opérateurs de différence finie
$\mathbb{Q}(x)\langle\partial;\mathrm{H},0\rangle \quad (q\in\mathbb{Q}\setminus\{0,1,-1\})$	opérateurs de q-dilatation
$\mathbb{Q}(x)\langle\partial;\mathrm{id},0\rangle$	polynômes commutatifs usuels de $\mathbb{Q}(x)[\partial]$

Figure 30.1 – Algèbres d'opérateurs linéaires représentés par les anneaux de polynômes tordus les plus courants.

Anneau	Commutation avec x	Commutation avec $u\in\mathbb{Q}(x)$
$\mathbb{Q}(x)\langle\partial;\mathrm{id},\mathrm{D}\rangle$	$\partial x = x\partial + 1$	$\partial u = u\partial + u'$
$\mathbb{Q}(x)\langle\partial;\mathrm{S},0\rangle$	$\partial x = (x+1)\partial$	$\partial u = u(x+1)\partial$
$\mathbb{Q}(x)\langle\partial;\mathrm{S},\Delta\rangle$	$\partial x = (x+1)\partial + 1$	$\partial u = u(x+1)\partial + u(x+1) - u(x)$
$\mathbb{Q}(x)\langle\partial;\mathrm{H},0\rangle$	$\partial x = qx\partial$	$\partial u = u(qx)\partial$
$\mathbb{Q}(x)\langle\partial;\mathrm{id},0\rangle$	$\partial x = x\partial$	$\partial u = u\partial$

Figure 30.2 – Règles de commutation pour les anneaux de polynômes tordus les plus courants.

nous noterons 1 l'élément neutre pour le produit de polynômes tordus. Néanmoins, nous ferons l'abus de notation de noter la dérivation par rapport à x de la même façon, D_x, quel que soit l'anneau A, et id, sans indice, pour l'identité de n'importe quel A. De plus, nous noterons simplement x pour M. Nous obtenons ainsi les règles de commutations données explicitement en Figure 30.2.

Le cas $\delta = 0$ est fréquent, et on écrit alors $\mathrm{A}\langle\partial;\sigma\rangle$, sans référence au 0. De même que dans le cas commutatif on définit les polynômes de Laurent, dont l'algèbre est notée $\mathrm{A}[\mathrm{X},\mathrm{X}^{-1}]$, et dans laquelle $\mathrm{X}\mathrm{X}^{-1} = \mathrm{X}^{-1}\mathrm{X} = 1$, le cas où σ est inversible et autre que l'identité permet de représenter des opérateurs qui possèdent un inverse. Dans ce cas, on notera $\mathrm{A}\langle\partial,\partial^{-1};\sigma\rangle$ l'algèbre où $\partial a = \sigma(a)\partial$, $\partial^{-1}a = \sigma^{-1}(a)\partial^{-1}$, $\partial\partial^{-1} = \partial^{-1}\partial = 1$.

Idéaux, modules, actions

Les anneaux de polynômes tordus ont été définis par leur règle de commutation, sans faire référence à quelque opérateur que ce soit. Leur interprétation comme algèbres d'opérateurs, ainsi que les propriétés des calculs dans les sections et chapitres qui suivent, s'expriment mieux en termes de quelques structures algébriques non commutatives que nous définissons dans cette section.

Les structures algébriques qui vont servir se placent dans le cadre des anneaux généraux, c'est-à-dire, non commutatifs : un *anneau* R est la donnée d'un ensemble muni d'une première loi interne + appelée *addition*, qui en fait un groupe commutatif de neutre 0, ainsi que d'une deuxième loi interne appelée *multiplication* et dénotée par juxtaposition, qui en fait un monoïde de neutre 1 et est distributive à gauche et à droite sur l'addition. Les « anneaux commutatifs » déjà utilisés dans les chapitres précédents et dans la Définition 30.1 des « anneaux de polynômes tordus » sont des

anneaux (au sens général qui vient d'être indiqué) dont on exige que la multiplication soit commutative. Un anneau de polynômes tordus est bien un anneau particulier.

Tout comme les anneaux commutatifs ont des idéaux, définis comme des sous-ensembles clos par multiplication à gauche et à droite par tout élément de l'anneau, les anneaux généraux ont des idéaux, mais on ne s'intéresse en général qu'à la clôture d'un unique côté. Un *idéal à gauche* I d'un anneau R est un sous-groupe de R (pour l'addition) tel que $fg \in \mathrm{I}$ pour tout $f \in \mathrm{R}$ et tout $g \in \mathrm{I}$. Une notion d'idéal à droite pourrait être définie de façon analogue ; dans tout ce qui suit, nous dirons simplement idéal pour idéal à gauche. Pour une famille finie $g_1, \ldots, g_s$ d'éléments de R, les sommes $f_1 g_1 + \cdots + f_s g_s$ pour des coefficients $f_i \in \mathrm{R}$ constituent un idéal noté $\mathrm{R}g_1 + \cdots + \mathrm{R}g_s$, appelé *idéal engendré par* $g_1, \ldots, g_s$. Tout anneau R peut être vu comme idéal de lui-même car $\mathrm{R} = \mathrm{R}1$.

De même, la notion de module sur un anneau commutatif, notion déjà utilisée pour décrire les approximants de Padé–Hermite (Chapitre 7), la réduction de réseau (Chapitre 20), et dans toute la partie sur les systèmes polynomiaux, se généralise au cadre des anneaux généraux. Un *module à gauche* M sur un anneau R est un groupe additif muni d'une addition + et d'un neutre 0, stable sous l'action d'une loi externe $\cdot$ de multiplication à gauche par les éléments de R, appelée *multiplication scalaire*, distributive à gauche sur l'addition de R et à droite sur l'addition de M, et telle que pour tous r et s de R et m de M, $(rs) \cdot m = r \cdot (s \cdot m)$ et $1 \cdot m = m$. Une notion de module à droite pourrait être définie de façon analogue ; dans tout ce qui suit, nous dirons simplement module pour module à gauche. Cette notion généralise celle d'idéal : tout idéal I de R est un module sur R, obtenu en prenant la multiplication de R comme multiplication scalaire. Pour une famille finie $g_1, \ldots, g_s$ d'éléments d'un module M fixé, les sommes $f_1 g_1 + \cdots + f_s g_s$ pour des coefficients $f_i \in \mathrm{R}$ constituent un module S noté $\mathrm{R}g_1 + \cdots + \mathrm{R}g_s$, appelé *module engendré par* $g_1, \ldots, g_s$. C'est un *sous-module* de M sur R, au sens où c'est un module sur R obtenu par restriction à S de l'addition de M et de la multiplication scalaire. On appelle encore *action de* R *sur* M la loi de multiplication scalaire d'un module M.

Pour un module donné M sur R, soit G une famille $(g_i)_{i \in \mathrm{I}}$ d'éléments de M. La famille G est un *système de générateurs* de M si pour tout $m \in \mathrm{M}$, il existe une sous-famille finie $g'_1, \ldots, g'_s$ de G et des coefficients $f_i \in \mathrm{R}$ tels que $m = f_1 g'_1 + \cdots + f_s g'_s$. La famille G est *libre* si toute relation de dépendance linéaire $f_1 g'_1 + \cdots + f_s g'_s = 0$ obtenue pour une sous-famille finie $g'_1, \ldots, g'_s$ de G implique que tous les f_i sont nuls. La famille G est une *base* de M si c'est un système de générateurs et si elle est libre. Un module est dit *libre* s'il possède une base. Pour les anneaux R qui nous intéressent, à savoir les anneaux commutatifs et les anneaux de polynômes tordus à coefficients dans un anneau commutatif, si un module libre sur R possède une base, le cardinal de cette base est indépendant du choix de la base. Dans le cas de bases finies, ce cardinal est appelé *rang* du module. Un cas particulier important est celui du produit cartésien R^n pour $n \in \mathbb{N}$, qui fournit un module pour l'addition coordonnée à coordonnée et pour la multiplication scalaire obtenue en multipliant chaque coordonnée par le même élément de R. C'est un module libre de rang n, de base constituée par les éléments $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$, où l'unique 1 est en position i.

Pour relier anneaux de polynômes tordus et algèbres d'opérateurs, nous faisons maintenant agir les anneaux de polynômes tordus sur les espaces de fonctions, au

sens de l'action d'un anneau sur un module. Mais un anneau de polynômes tordus n'a pas d'action unique sur un espace de fonctions donné, aussi adoptons-nous les conventions, en vigueur dans toute la suite du texte :

— un anneau de la forme $A\langle\partial;\sigma\rangle$ agit par $\partial\cdot f=\sigma(f)$ pour une extension convenable de σ,
— un anneau de la forme $A\langle\partial;\sigma,\delta\rangle$ agit par $\partial\cdot f=\delta(f)$ pour une extension convenable de δ,
— les coefficients dans A agissent par simple multiplication, $a\cdot f=af$.

Remarquons que l'algèbre des suites $A^{\mathbb{N}}$ ne peut pas être vue comme un module sur l'anneau $A\langle\partial,\partial^{-1};\sigma\rangle$ quand σ est le décalage avant. En effet, l'action de ∂ sur la suite valant 1 en 0 et nulle ensuite donne la suite nulle : l'action de ∂ ne peut donc pas être inversée. Ce problème technique peut être contourné en considérant les classes de suites identifiées lorsqu'elles sont égales après un certain rang (qui dépend des deux suites). (On parle de « germe de suite à l'infini ».) Mais on perd alors la possibilité d'exprimer certaines suites, celles à support fini, justement.

30.2 Morphismes entre anneaux de polynômes tordus

Dans cette section, nous sommes amenés à considérer simultanément des fonctions de x et des fonctions d'une autre variable. Aussi indiquerons-nous en indice de D, S, ∂, σ, δ, etc, la variable à laquelle ces objets font référence. De plus, les anneaux A qui servent à construire les anneaux de polynômes tordus sont de la forme $\mathbb{Q}[x]$ ou $\mathbb{Q}[x,x^{-1}]$.

Série génératrice différentiellement finie et suite polynomialement récurrente associée

On a déjà vu que lorsqu'une série $f=\sum_{n\geq 0}u_nx^n$ est différentiellement finie, ses coefficients vérifient une relation de récurrence finie. Autrement dit, la suite $u=(u_n)_{n\geq 0}$ est polynomialement récurrente. La preuve repose sur les identités

$$xf=\sum_{n\geq 1}u_{n-1}x^n=\sum_{n\geq 1}(\partial_n^{-1}\cdot u)(n)\,x^n$$

et

$$\mathrm{D}_x(f)=f'=\sum_{n\geq 0}(n+1)u_{n+1}x^n=\sum_{n\geq 0}\Big((n+1)\partial_n\cdot u\Big)(n)\,x^n,$$

où nous avons introduit l'anneau $\mathbb{Q}[n]\langle\partial_n,\partial_n^{-1};\mathrm{S}_n\rangle$. Par récurrence, ceci donne

$$x^\alpha\mathrm{D}_x^\beta(f)=\sum_{n\geq\alpha}\Big(\partial_n^{-\alpha}\Big((n+1)\partial_n\Big)^\beta\cdot u\Big)(n)\,x^n=\sum_{n\geq\alpha}\Big((n+1-\alpha)\cdots(n+\beta-\alpha)\partial_n^{\beta-\alpha}\cdot u\Big)(n)\,x^n.$$

Pour une série f solution de l'équation différentielle

$$a_r(x)f^{(r)}(x)+\cdots+a_0(x)f(x)=0$$

où les a_i sont dans $\mathbb{Q}[x]$, nous obtenons ainsi une récurrence sur u, valable pour des n assez grands. Cette récurrence s'exprime en termes de polynômes tordus de la façon suivante. On représente l'opérateur différentiel associé à l'équation par le polynôme tordu $\mathrm{L} = a_r(x)\partial_x^r + \cdots + a_0(x)$ dans $\mathbb{Q}[x]\langle\partial_x; \mathrm{id}, \mathrm{D}_x\rangle$ sur $\mathbb{Q}$. De la sorte, l'équation différentielle s'écrit $\mathrm{L}\cdot f = 0$. On introduit aussi l'algèbre $\mathbb{Q}[n]\langle\partial_n, \partial_n^{-1}; \mathrm{S}_n\rangle$ et le morphisme d'algèbres μ défini par $\mu(x) = \partial_n^{-1}$ et $\mu(\partial_x) = (n+1)\partial_n$. Alors, la suite u des coefficients est solution pour n assez grand de la récurrence représentée par l'image $\mu(\mathrm{L})$. Pour comprendre pour quels n cette récurrence est valide, écrivons

$$\mu(\mathrm{L}) = b_p(n)\partial_n^p + \cdots + b_q(n)\partial_n^q$$

pour $p \le q$ et $b_p b_q \neq 0$, de sorte que la récurrence prend la forme

$$\Big(\mu(\mathrm{L})\cdot u\Big)(n) = b_p(n)u_{n+p} + \cdots + b_q(n)u_{n+q} = 0$$

et est vérifiée pour tout n si $p \ge 0$ et pour tout $n \ge -p$ si $p < 0$.

De façon duale, une suite polynomialement récurrente u a une série génératrice $f = \sum_{n\ge 0} u_n x^n$ différentiellement finie, ce que nous allons retrouver en termes de polynômes tordus. Pour ce point, nous supposons en fait que la suite u est prolongée aux indices négatifs par $u_n = 0$ pour $n < 0$, et qu'elle est polynomialement récurrente sur $\mathbb{Z}$ tout entier. Ceci ne constitue aucune perte de généralité : une récurrence valable pour la suite initiale devient valable pour la suite prolongée après multiplication par un polynôme de la forme $(n+1)(n+2)\cdots(n+r)$. Les formules

$$\sum_{n\in\mathbb{Z}} n u_n x^n = x\partial_x \cdot f \qquad \text{et} \qquad \sum_{n\in\mathbb{Z}} u_{n+1} x^n = x^{-1} f$$

donnent par récurrence

$$\sum_{n\ge 0} n^\alpha u_{n+\beta} x^n = (x\partial_x)^\alpha x^{-\beta}\cdot f = x^{-\beta}(x\partial_x - \beta)^\alpha \cdot f,$$

et fournissent un autre morphisme, ν, de $\mathbb{Q}[n]\langle\partial_n; \mathrm{S}_n\rangle$ dans $\mathbb{Q}[x, x^{-1}]\langle\partial_x; \mathrm{id}, \mathrm{D}_x\rangle$, donné par $\nu(n) = x\partial_x$ et par $\nu(\partial_n) = x^{-1}$. Pour une suite u solution sur $\mathbb{Z}$ de l'équation de récurrence

$$b_p(n)u_{n+p} + \cdots + b_0(n)u_n = 0$$

où les b_i sont dans $\mathbb{Q}[n]$, nous introduisons le polynôme tordu $\mathrm{P} = b_p(n)\partial_n^p + \cdots + b_0(n)$ de $\mathbb{Q}[n]\langle\partial_n; \mathrm{S}_n\rangle$. Pour obtenir une relation différentielle sur la série génératrice f, nous considérons $\nu(\mathrm{P})$ que nous écrivons

$$\nu(\mathrm{P}) = a_0(x) + \cdots + a_r(x)\partial_x^r.$$

Alors, comme $\sum_{n\in\mathbb{Z}}(\mathrm{P}\cdot u)(n)x^n = 0$, la série f vérifie la relation différentielle

$$a_0(x)f(x) + \cdots + a_r(x)f^{(r)}(x) = 0.$$

Algébriquement, les propriétés précédentes s'expriment par le fait que μ s'étend en un isomorphisme d'algèbres entre $\mathbb{Q}[x, x^{-1}]\langle\partial_x; \mathrm{id}, \mathrm{D}_x\rangle$ et $\mathbb{Q}[n]\langle\partial_n, \partial_n^{-1}; \mathrm{S}_n\rangle$, dont l'inverse étend ν. Ce morphisme vérifie $\mu(x^i) = \partial_n^{-i}$ et

$$\mu(\partial_x^i) = \Big((n+1)\partial_n\Big)^i = (n+1)\cdots(n+i)\partial_n^i.$$

Les deux algèbres isomorphes peuvent être vues comme agissant naturellement sur $\mathbb{Q}((x)) = \{x^{-r}f \mid r \in \mathbb{N},\ f \in \mathbb{Q}[[x]]\}$, qui n'est autre que l'algèbre des suites sur $\mathbb{Z}$ nulles pour les indices n strictement inférieurs à un entier $-r$ (dépendant de la suite considérée).

Séries binomiales

Exercice 30.1 Une série binomiale est une série de la forme $\sum_{n\geq 0} u_n\binom{x}{n}$. Montrer que les solutions en série binomiale d'une équation fonctionnelle à différence

$$a_r(x)f(x+r) + \cdots + a_0(x)f(x) = 0$$

ont des coefficients u_n qui vérifient une récurrence et expliciter le morphisme entre algèbres de polynômes tordus correspondant. ∎

Changements de variables

Lorsqu'une série différentiellement finie $f(x)$ est solution d'une équation différentielle $\mathrm{L}\cdot f = 0$ donnée par un polynôme tordu

$$\mathrm{L} = \mathrm{L}(x, \partial_x) = a_r(x)\partial_x^r + \cdots + a_0(x),$$

pour toute constante $\lambda \neq 0$ la série $g(x) := f(\lambda x)$ vérifie $g^{(i)}(x) = \lambda^i f^{(i)}(\lambda x)$ pour chaque $i \in \mathbb{N}$. En substituant λx pour x dans $\mathrm{L}\cdot f = 0$, on obtient que g est solution de l'équation différentielle associée à

$$\mathrm{L}(\lambda x, \lambda^{-1}\partial_x) = a_r(\lambda x)\lambda^{-r}\partial_x^r + \cdots + a_0(\lambda x).$$

C'est encore le résultat d'un morphisme d'algèbres, Λ, défini cette fois par $\Lambda(x) = \lambda x$ et $\Lambda(\partial_x) = \lambda^{-1}\partial_x$.

Lorsque f est une fonction différentiellement différentiellement finie, la fonction $z \mapsto f(1/z)$ est elle aussi différentiellement finie, en z cette fois, pour autant que la fonction composée ait un sens. En effet, pour toute fonction g, notons $\tilde{g}(z) = g(1/z)$ (avec la même réserve de définition). Puisque $g(x) = \tilde{g}(1/x)$, par dérivation on a $g'(x) = -\tilde{g}'(1/x)/x^2$, ce qui est l'évaluation en $z = 1/x$ de $-z^2\partial_z \cdot \tilde{g}$. Autrement dit, on a $\widetilde{g'} = -z^2\partial_z \cdot \tilde{g}$, d'où par récurrence $\widetilde{g^{(\beta)}} = (-z^2\partial_z)^\beta \cdot \tilde{g}$. Ainsi, $\tilde{f}$ est différentiellement finie, donnée comme vérifiant l'équation différentielle associée à l'image de L par le morphisme de $\mathbb{Q}[x]\langle\partial_x; \mathrm{id}, \mathrm{D}_x\rangle$ dans $\mathbb{Q}[z, z^{-1}]\langle\partial_z; \mathrm{id}, \mathrm{D}_z\rangle$ qui envoie x sur z^{-1} et ∂_x sur $-z^2\partial_z$.

Exercice 30.2 Plus généralement, la fonction obtenue par substitution rationnelle de la variable, donnée par $h(u) = f\big(r(u)\big)$, est encore différentiellement finie. Nous laissons en exercice le soin de montrer ce résultat par la même approche dans le cas où la dérivée r' s'exprime comme une fraction rationnelle en r. ∎

30.3 Division euclidienne

Dans cette section et les suivantes, nous nous appuyons sur des propriétés particulières des anneaux de polynômes tordus quand l'anneau A de la construction est un

corps, que nous prendrons de la forme $\mathbb{Q}(x)$.

La commutation $\partial a = \sigma(a)\partial + \delta(a)$ dans $\mathbb{Q}(x)\langle\partial;\sigma,\delta\rangle$ permet d'écrire tout polynôme tordu sous la forme $a_0(x) + \cdots + a_r(x)\partial^r$, pour des fractions rationnelles a_i de $\mathbb{Q}(x)$ uniques. Une conséquence de l'injectivité de σ est l'existence d'un degré en ∂ bien défini : l'entier r de l'écriture précédente lorsque a_r est non nulle. En particulier, le degré d'un produit $\mathrm{L}_1\mathrm{L}_2$ de polynômes tordus est la somme des degrés des L_i. Il s'ensuit que la division euclidienne du cas commutatif, et toute la théorie qui en découle, se transpose avec peu d'altérations dans le cas tordu.

Procédé de division

La différence principale avec le cas commutatif est qu'on distingue division euclidienne à gauche et division euclidienne à droite. Vu notre interprétation en termes d'opérateurs linéaires, nous ne considérerons que la division à droite, qui se fait en retranchant des multiples à gauche. Soit à diviser $\mathrm{A} = a_r(x)\partial^r + \cdots + a_0(x)$ de degré r par $\mathrm{B} = b_s(x)\partial^s + \cdots + b_0(x)$ de degré s. On suppose $s \leq r$. Alors,

$$\partial^{r-s}\mathrm{B} = \sigma^{r-s}\Big(b_s(x)\Big)\partial^r + \text{termes d'ordres inférieurs},$$

où la puissance de σ représente une itération (par composition), et ainsi

$$\mathrm{A} - a_r(x)\sigma^{r-s}\Big(b_s(x)\Big)^{-1}\partial^{r-s}\mathrm{B}$$

est de degré strictement inférieur à r. Cette étape de réduction est l'étape élémentaire de la division euclidienne. En itérant le procédé, on aboutit à un reste R de degré strictement inférieur à s. En regroupant les facteurs gauches, on obtient un quotient à gauche Q tel que $\mathrm{A} = \mathrm{QB} + \mathrm{R}$.

Exemple 30.1 On considère l'anneau $\mathbb{Q}(n)\langle\partial_n;\mathrm{S}_n\rangle$ des polynômes tordus représentant les opérateurs de décalage. La division de $\mathrm{A} = (n^2-1)\partial_n^2 - (n^3+3n^2+n-2)\partial_n + (n^3+3n^2+2n)$, qui annule les combinaisons linéaires de $n!$ et n, par $\mathrm{B} = n\partial_n^2 - (n^2+3n+1)\partial_n + (n^2+2n+1)$, qui annule les combinaisons linéaires de $n!$ et 1, s'écrit

$$\mathrm{A} = n^{-1}(n^2-1)\mathrm{B} - n^{-1}(n^2+n+1)\Big(\partial_n - (n+1)\Big).$$

Le reste est multiple de $\partial_n - (n+1)$, qui représente la récurrence $u_{n+1} = (n+1)u_n$, vérifiée par la factorielle.

Notons une propriété de cette division : si A est multiplié à gauche par un facteur $m(x)$ sans que B ne soit changé, alors Q et R sont multipliés à gauche par le même facteur $m(x)$. Ceci ne vaut plus (en général) pour un facteur faisant intervenir ∂.

Réduction d'une puissance de ∂

La division euclidienne nous donne une nouvelle interprétation du calcul du N-ième terme d'une suite polynomialement récurrente $u = (u_n)$ relativement à l'anneau $\mathbb{Q}(n)\langle\partial_n;\mathrm{S}_n\rangle$. Supposons que u soit solution de l'équation de récurrence

$$a_r(n)u_{n+r} + \cdots + a_0(n)u_n = 0. \tag{30.1}$$

En déroulant la récurrence, on voit que u_{N} peut, pour tout N sauf annulation malvenue de a_r, se mettre sous la forme $\alpha_{r-1,\mathrm{N}}u_{r-1} + \cdots + \alpha_{0,\mathrm{N}}u_0$. Plus généralement, on a une relation qui récrit $u_{n+\mathrm{N}}$ en termes de $u_{n+r-1}, \ldots, u_n$. Pour l'obtenir, associons à la récurrence sur u le polynôme tordu $\mathrm{P} = a_r(n)\partial_n^r + \cdots + a_0(n)$. Pour un N donné, la division euclidienne de ∂_n^{N} par P s'écrit

$$\partial_n^{\mathrm{N}} = \mathrm{Q}_{\mathrm{N}}(n)\mathrm{P} + \alpha_{r-1,\mathrm{N}}(n)\partial_n^{r-1} + \cdots + \alpha_{0,\mathrm{N}}(n)$$

pour des fractions rationnelles $\alpha_{i,\mathrm{N}}(n)$. Après application sur u et évaluation en n, nous obtenons

$$u_{n+\mathrm{N}} = 0 + \alpha_{r-1,\mathrm{N}}(n)u_{n+r-1} + \cdots + \alpha_{0,\mathrm{N}}(n)u_n,$$

d'où le résultat annoncé pour $\alpha_{i,\mathrm{N}} = \alpha_{i,\mathrm{N}}(0)$.

Exemple 30.2 Dans le cas particulier d'une relation de récurrence (30.1) d'ordre 1, définissant donc une suite $u(n)$ hypergéométrique, nous obtenons par linéarité la relation $(\mathrm{L}\cdot u)(n) = r(n)u(n)$, où la fraction rationnelle $r(n)$ est le reste de la division euclidienne de L par $\partial_n - a_0(n)/a_1(n)$. Nous retrouvons ainsi le Lemme 29.1.

Exercice 30.3 Nous laissons le lecteur se convaincre que la récriture d'une dérivée $f^{(\mathrm{N})}$ d'une fonction différentiellement finie f décrite par une équation différentielle d'ordre r en termes de ses dérivées d'ordre strictement inférieur à r s'interprète de façon analogue comme le calcul d'un reste de division euclidienne. ∎

Clôture par addition

Le même ingrédient se retrouve dans l'algorithme donnant la clôture par addition de deux fonctions différentiellement finies ou de deux suites polynomialement récurrentes : pour deux objets f et g à additionner, décrits comme solutions des équations respectives $\mathrm{L}_f \cdot f = 0$ et $\mathrm{L}_g \cdot g = 0$ pour des polynômes tordus de degrés respectifs r et s dans un anneau adéquat $\mathrm{A}\langle\partial;\sigma,\delta\rangle$, l'algorithme exprime, pour des i successifs, $\partial^i \cdot (f+g)$ sous la forme $(\partial^i \bmod \mathrm{L}_f)\cdot f + (\partial^i \bmod \mathrm{L}_g)\cdot g$, où la notation A mod B note le reste de la division euclidienne à droite de A par B. Lorsque suffisamment de i ont été considérés, l'algorithme qui a été donné au Chapitre 14 calcule par de l'algèbre linéaire des cofacteurs $a_0, \ldots, a_{r+s}$ tels que, simultanément,

$$\sum_{i=0}^{r+s} a_i(\partial^i \bmod \mathrm{L}_f) = 0 \qquad \text{et} \qquad \sum_{i=0}^{r+s} a_i(\partial^i \bmod \mathrm{L}_g) = 0.$$

Notons que $\mathrm{P} = \sum_{i=0}^{r+s} a_i\partial^i$ est un multiple commun à gauche de L_f et de L_g, puisque $\mathrm{P} \bmod \mathrm{L}_f = \mathrm{P} \bmod \mathrm{L}_g = 0$. Puisque l'algorithme fonctionne en recherchant un P d'ordre minimal, l'algorithme de clôture par addition est donc un algorithme de calcul de ppcm (à gauche).

30.4 Recherche de solutions et factorisation d'opérateurs

Comme pour les anneaux de polynômes commutatifs usuels, une notion de factorisation est présente pour les anneaux de polynômes tordus. Une nuance importante

réside dans le lien entre les « zéros » des polynômes tordus et la position des facteurs. Nous allons voir que la factorisation de polynômes tordus se relie aux algorithmes vus dans cet ouvrage pour la recherche de solutions polynomiales, rationnelles, et hypergéométriques dans le cas de récurrences. Par ailleurs, il n'y a plus d'unicité des facteurs, même à ordre près et modulo multiplication par des unités. Un exemple est donné dans le cas différentiel par l'infinité de factorisations

$$\partial^2 = \left(\partial + \frac{1}{x+r}\right)\left(\partial - \frac{1}{x+r}\right)$$

quand r est une constante.

Dans le cas d'un polynôme commutatif w se factorisant sous la forme uv pour des facteurs polynomiaux de degré au moins 1, tout zéro de u et tout zéro de v est zéro de w ; à l'inverse, quitte à se placer dans une clôture algébrique, tout zéro α de w en fournit un facteur $x - \alpha$ et un quotient exact $u(x)$ tel que $w(x) = u(x)(x - \alpha)$. Le phénomène est différent dans le cas tordu, comme attesté par l'inégalité

$$\partial^2 \neq \left(\partial - \frac{1}{x+r}\right)\left(\partial + \frac{1}{x+r}\right)$$

quand r est une constante.

Plus généralement, une factorisation $\mathrm{L} = \mathrm{PQ}$ dans $\mathrm{A}\langle\partial;\sigma,\delta\rangle$ (où A est un corps) a des propriétés différentes selon le facteur : une solution f de l'équation $\mathrm{Q}\cdot f = 0$ est encore solution de $\mathrm{L}\cdot f = 0$, car $\mathrm{L}\cdot f = \mathrm{P}\cdot(\mathrm{Q}\cdot f) = \mathrm{P}\cdot 0 = 0$; mais une solution g de P ne donne lieu à des solutions f de L que par la relation $\mathrm{Q}\cdot f = g$. Inversement, une solution f de L donne lieu à un facteur droit d'ordre 1 de L, à condition d'étendre A par f et tous ses itérés par σ et δ. Ce facteur est de la forme $\partial - (\partial\cdot f)/f$, c'est-à-dire $\partial - \delta(f)/f$ ou $\partial - \sigma(f)/f$ selon l'action de l'anneau de polynômes tordus sur les fonctions.

Les algorithmes de recherche de solutions dans des classes particulières fournissent donc implicitement, pour chaque solution trouvée, un facteur droit d'ordre 1. Ils interviennent donc naturellement comme base d'algorithmes de factorisation. Plus précisément, dans le cas différentiel, une solution polynomiale ou rationnelle f d'une équation $\mathrm{L}\cdot f = 0$ pour L dans l'anneau $\mathbb{Q}(x)\langle\partial_x;\mathrm{id},\mathrm{D}_x\rangle$ fournit un facteur droit $\mathrm{D} = \partial_x - (\mathrm{D}_x\cdot f)/f$ où $(\mathrm{D}_x\cdot f)/f$ est rationnel ; dans le cas à récurrence, une solution polynomiale, rationnelle ou hypergéométrique f d'une équation $\mathrm{L}\cdot f = 0$ pour L dans l'anneau $\mathbb{Q}(x)\langle\partial_x;\mathrm{S}_x\rangle$ fournit un facteur droit $\mathrm{D} = \partial_x - \mathrm{S}_x(f)/f$ où $\mathrm{S}_x(f)/f$ est rationnel. Dans les deux cas, le quotient Q tel que $\mathrm{L} = \mathrm{QD}$ est aussi à coefficients rationnels.

Exemple 30.3 Un exemple simple, en réalité sans extension stricte de A, est donné dans $\mathrm{A} = \mathbb{Q}[x]\langle\partial_x;\mathrm{id},\mathrm{D}_x\rangle$ par l'opérateur

$$\mathrm{L} = x^2(4x^2+5)\partial_x^3 - x(8x^4+14x^2-5)\partial_x^2 - (4x^4+17x^2+5)\partial_x - 2x^3(4x^2+9),$$

dont une solution particulière est $f = \exp x^2$. Cette solution se vérifie à partir de

$\mathrm{D}_x \cdot f = 2xf$, $\mathrm{D}_x^2 \cdot f = (4x^2+2)f$ et $\mathrm{D}_x^3 \cdot f = (8x^3+12x)f$, en établissant la nullité de

$$x^2(4x^2+5)(8x^3+12x) - x(8x^4+14x^2-5)(4x^2+2) - (4x^4+17x^2+5)(2x) - 2x^3(4x^2+9).$$

Par division euclidienne, on trouve

$$\mathrm{L} = \Big(x^2(4x^2+5)\partial_x^2 - x(4x^2-5)\partial_x + 4x^4 + 13x^2 - 5\Big)(\partial_x - 2x).$$

On montre que L n'a pas d'autre facteur droit non trivial unitaire ; l'Exercice 30.4 fournit les facteurs gauches d'ordre 1, en réalité un seul sur cet exemple :

$$\mathrm{L} = \Big(x(4x^2+5)\partial_x - (2x^2+5)(4x^2+1)\Big)(x\partial_x^2 + \partial_x + x).$$

Exemple 30.4 L'opérateur $\mathrm{L} = 2(2x+3)\partial_x^2 + 3(5x+7)\partial_x + 9(x+1)$ fournit un autre exemple, pour $\mathrm{A} = \mathbb{Q}[x]\langle\partial_x;\mathrm{S}_x\rangle$. Comme une solution particulière est $(-3)^x$, ce qui s'obtient par l'algorithme de Petkovšek et peut s'établir simplement en observant la nullité de L en $\partial_x = -3$, un facteur droit de L est $\partial_x + 3$, et une division euclidienne montre

$$\mathrm{L} = \Big(2(2x+3)\partial_x + 3(x+1)\Big)(\partial_x + 3).$$

On montre que L n'a pas d'autre facteur droit non trivial unitaire.

Exercice 30.4 Un antimorphisme ϕ entre anneaux est une application $\mathbb{Q}$-linéaire qui renverse les produits : $\phi(\mathrm{PQ}) = \phi(\mathrm{Q})\phi(\mathrm{P})$. Montrer l'existence d'antimorphismes

$$\mu : \mathbb{Q}(x)\langle\partial_x;\mathrm{id},\mathrm{D}_x\rangle \to \mathbb{Q}(u)\langle\partial_u;\mathrm{id},\mathrm{D}_u\rangle$$

et

$$\nu : \mathbb{Q}(x)\langle\partial_x;\mathrm{S}_x\rangle \to \mathbb{Q}(u)\langle\partial_u;\mathrm{S}_u\rangle,$$

définis par les relations

$$\mu(x) = u, \quad \mu(\partial_x) = -\partial_u, \qquad \text{et} \qquad \nu(x) = -u, \quad \nu(\partial_x) = \partial_u.$$

Expliquer comment ces antimorphismes fournissent des facteurs gauches d'ordre 1 de polynômes tordus. ■

30.5 Algorithme d'Euclide

Il est classique que les anneaux commutatifs euclidiens — ceux dans lesquels l'existence d'un degré permet une division euclidienne — sont principaux : tout idéal peut être engendré par un unique générateur. C'est le cas des anneaux de polynômes commutatifs à coefficients dans un corps. Le pgcd p de deux polynômes f et g est alors l'unique polynôme unitaire engendrant l'idéal (f,g), somme des idéaux (f) et (g). Il se calcule comme dernier reste non nul par l'algorithme d'Euclide.

Pour un anneau de polynômes tordus $\mathrm{A} = \mathrm{K}\langle\partial;\sigma,\delta\rangle$ sur un corps K, la situation est la même si on prend soin de ne considérer que des idéaux à gauche, c'est-à-dire

Entrée F et F′ dans $A = K\langle\partial; \sigma, \delta\rangle$.
Sortie Un pgcdd G de F et F′, les cofacteurs U et V d'une relation de Bézout $G = UF + VF'$, des cofacteurs R et S donnant un ppcmg $RF = -SF'$.

1. $P_0 := F$; $U_0 := 1$; $V_0 := 0$; $P_1 := F'$; $U_1 := 0$; $V_1 := 1$; $i := 1$.
2. Tant que P_i est non nul :
   a. calculer le quotient Q_{i-1} et le reste P_{i+1} de la division euclidienne à droite de P_{i-1} par P_i, de sorte que $P_{i-1} := Q_{i-1}P_i + P_{i+1}$;
   b. $U_{i+1} := U_{i-1} - Q_iU_i$; $V_{i+1} := V_{i-1} - Q_iV_i$;
   c. $i := i + 1$.
3. Renvoyer $G = R_{i-1}$, $(U, V) = (U_{i-1}, V_{i-1})$ et $(R, S) = (U_i, V_i)$.

Algorithme 30.3 – Algorithme d'Euclide étendu dans un anneau de polynômes tordus.

avec la clôture par multiplication à gauche par les éléments de A. Les notions qui en découlent sont celles de divisions euclidiennes à droite et de plus grands communs diviseurs à droite (pgcdd). Soient P_0 et P_1 deux polynômes de A. Si P_1 n'est pas nul, on écrit la division euclidienne de P_0 par P_1, sous la forme $P_0 = Q_0P_1 + P_2$. Tant que P_{i+2} n'est pas nul, on itère en divisant P_{i+1} par P_{i+2}. Soit j la valeur finale de i, telle que $P_{j+1} \neq 0$ et $P_{j+2} = 0$. Alors :

$$\begin{pmatrix} P_0 \\ P_1 \end{pmatrix} = \begin{pmatrix} Q_0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} P_1 \\ P_2 \end{pmatrix} = \cdots = \begin{pmatrix} Q_0 & 1 \\ 1 & 0 \end{pmatrix}\cdots\begin{pmatrix} Q_j & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} P_{j+1} \\ 0 \end{pmatrix},$$

d'où on déduit que P_{j+1} divise P_0 et P_1 à droite : $P_0 = FP_{j+1}$ et $P_1 = GP_{j+1}$ pour des polynômes tordus F et G adéquats. Puis en inversant les matrices

$$\begin{pmatrix} P_{j+1} \\ 0 \end{pmatrix} = \begin{pmatrix} U & V \\ R & S \end{pmatrix}\begin{pmatrix} P_0 \\ P_1 \end{pmatrix} \qquad \text{pour} \qquad \begin{pmatrix} U & V \\ R & S \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -Q_j \end{pmatrix}\cdots\begin{pmatrix} 0 & 1 \\ 1 & -Q_0 \end{pmatrix}.$$

En particulier, $P_{j+1} = UP_0 + VP_1$ est élément de l'idéal à gauche $AP_0 + AP_1$. Un élément quelconque $L = MP_0 + NP_1$ de cet idéal est aussi multiple de P_{j+1} : $L = (MF + NG)P_{j+1}$. En normalisant P_{j+1} pour le rendre unitaire, on obtient donc un pgcdd distingué de P_0 et P_1. Par ailleurs, le polynôme tordu $RP_0 = -SP_1$ est un plus petit multiple commun à gauche (ppcmg) de P_0 et P_1, par le même argument que dans le cas commutatif, en suivant de près les degrés tout au long de l'algorithme.

Cette discussion aboutit à l'Algorithme 30.3, que le lecteur comparera aux algorithmes d'Euclide et d'Euclide étendu du Chapitre 6, aux pages 113 et 116.

On a vu que les algorithmes de clôture par addition entre fonctions différentiellement finies ou entre suites polynomialement récurrentes renvoient un multiple commun à gauche des polynômes tordus L_f et L_g décrivant les deux objets f et g à additionner. Comme ces algorithmes opèrent par degrés croissants, le polynôme renvoyé est de degré minimal en ∂, parmi ceux qui annulent la somme $f + g$. Le polynôme annulateur de la somme $f + g$ renvoyé par ces algorithmes est donc le ppcmg de L_f et de L_g.

Exemple 30.5 Nous repartons des polynômes A et B de l'Exemple 30.1 pour en calculer un pgcdd et un ppcmg. On pose $P_0 = A$, $P_1 = B$; on a déjà calculé $P_2 = -n^{-1}(n^2+n+1)\big(\partial_n - (n+1)\big)$ avec $P_0 = n^{-1}(n^2-1)P_1 + P_2$. Il vient ensuite

$$P_1 = \left(-\frac{n(n+1)}{n^2+3n+3}\partial_n + \frac{n(n+1)}{n^2+n+1}\right)P_2 + 0.$$

Ainsi, le pgcdd unitaire est $\partial_n - (n+1)$. Remarquons qu'il annule les solutions communes de A et B, à savoir les multiples de $n!$. Le ppcmg unitaire s'obtient par renormalisation de $RP_0 = -SP_1$ quand $R = -Q_1$ et $S = Q_1Q_0$:

$$\partial_n^3 - \frac{n^3+6n^2+8n+5}{n^2+n+1}\partial_n^2 + \frac{2n^3+9n^2+13n+7}{n^2+n+1}\partial_n - \frac{(n^2+3n+3)(n+1)}{n^2+n+1}.$$

Notons que ses solutions sont toutes les solutions de A et B : les combinaisons linéaires de $n!$, n et 1.

30.6 Relations de contiguïté

Un grand nombre de fonctions spéciales sont en fait des fonctions $f_n(x)$ d'une variable continue x et d'une variable discrète n. Les familles de telles fonctions dont l'intérêt a été révélé, par exemple par la physique mathématique, sont telles que la dépendance en x est liée à la dépendance en n. Il apparaît que très fréquemment, la fonction $f_{n+\alpha}(x)$, pour $\alpha = \pm 1$, est reliée à la fonction $f_n(x)$ et à ses dérivées. C'est le cas pour la classe importante des *fonctions hypergéométriques*, c'est-à-dire, pour les fonctions de l'analyse données par les séries hypergéométriques introduites au Chapitre 29, Section 29.1, et pour des fonctions limites de fonctions hypergéométriques, dont un certain nombre de familles de polynômes orthogonaux classiques.

Dans cette section, nous considérons des fonctions differentiellement finies paramétrées et résolvons algorithmiquement des problèmes tels que la détermination d'une relation de la forme

$$f_{n+1}(x) = \sum_{i=0}^{r-1} a_i(x) f_n^{(i)}(x)$$

pour la suite de polynômes

$$f_n(x) = \sum_{k=0}^{n} (-1)^k \binom{n}{k}^2 \binom{n+k}{k}^2 x^k.$$

Ici, la relation explicite est

$$f_{n+1}(x) = 12\frac{x^3(x+1)}{(n+1)^3} f_n'''(x) + 4\frac{x^2(2n+2xn+11+14x)}{(n+1)^3} f_n''(x) - 4\frac{x(5xn^2-n^2-4n-6+2xn-9x)}{(n+1)^3} f_n'(x) - \frac{16xn-n-1+4x}{n+1} f_n(x).$$

L'opérateur différentiel linéaire implicitement au membre droit s'appelle un *opérateur de montée*; un opérateur qui donnerait $f_{n-1}(x)$ s'appelle un *opérateur de descente*. Une relation linéaire entre les decalées $f_{n+i}(x)$ et ne faisant intervenir aucune dérivation s'appelle une *relation de contiguïté*.

Fonction hypergéométrique de Gauss

La plus simple des fonctions hypergéométriques est la fonction hypergéométrique de Gauss, définie par

$$\mathrm{F}(a,b;c;x) = {}_2\mathrm{F}_1\left(\begin{matrix}a,b\\c\end{matrix}\middle|\, x\right) = \sum_{k\geq 0} \frac{(a)_k (b)_k}{(c)_k}\frac{x^k}{k!}. \tag{30.2}$$

La notation ${}_2\mathrm{F}_1$ a déjà été introduite au Chapitre 29, ainsi que le symbole de Pochhammer

$$(s)_n = s(s+1)\cdots(s+n-1) = \frac{\Gamma(s+n)}{\Gamma(s)},$$

lequel vérifie

$$\frac{(s)_{n+1}}{(s)_n} = s+n \qquad \text{et} \qquad \frac{(s+1)_n}{(s)_n} = \frac{s+n}{s}.$$

Oublions la dépendance en b et c du coefficient de x^k dans la somme (30.2), coefficient que nous notons $u_{a,k}$. Nous avons

$$u_{a,k+1} = \frac{(a+k)(b+k)}{(c+k)(k+1)}u_{a,k} \qquad \text{et} \qquad u_{a+1,k} = \left(\frac{k}{a}+1\right)u_{a,k}.$$

La première de ces identités nous fournit le polynôme tordu

$$(c+k)(k+1)\partial_k - (a+k)(b+k) \in \mathbb{Q}(a,b,c)[k]\langle\partial_k;\mathrm{S}_k\rangle$$

qui annule u. Pour $k=-1$, cette récurrence impose $u_{a,-1}=0$, ce qui permet d'étendre la suite u à toute valeur $k\in\mathbb{Z}$ tout en continuant de vérifier la récurrence. Par le morphisme vu en Section 30.2 pour effectuer le passage à la série génératrice, nous obtenons

$$\begin{aligned}(c+x\partial_x)(x\partial_x+1)x^{-1} &- (a+x\partial_x)(b+x\partial_x)\\ &= \Big((x\partial_x)^2 + (c+1)x\partial_x + c\Big)x^{-1} - \Big((x\partial_x)^2 + (a+b)x\partial_x + ab\Big)\\ &= x(1-x)\partial_x^2 + \Big(c-(a+b+1)x\Big)\partial_x - ab \in \mathbb{Q}(a,b,c)[x,x^{-1}]\langle\partial_x;\mathrm{id},\mathrm{D}_x\rangle.\end{aligned}$$

Notons L ce polynôme tordu en ∂_x. La deuxième identité sur u donne, après sommation, $\mathrm{F}(a+1,b;c;x) = (a^{-1}x\partial_x+1)\cdot\mathrm{F}(a,b;c;x)$; c'est-à-dire qu'un opérateur de montée est donné par le polynôme tordu $\mathrm{L}_\uparrow = a^{-1}x\partial_x + 1$.

Définissons $\mathrm{G}(a,b;c;x)$ comme étant $\mathrm{F}(a+1,b;c;x)$. Supposons qu'il existe un inverse à gauche V de $\mathrm{L}_\uparrow$ modulo L à droite. Alors $\mathrm{VL}_\uparrow - 1$ est un multiple à gauche de L, qui annule donc F. Ainsi, $\mathrm{F} = \mathrm{VL}_\uparrow\cdot\mathrm{F} = \mathrm{V}\cdot\mathrm{G}$, autrement dit, V représente un opérateur

de descente de G. On obtient un opérateur de descente pour F par un simple décalage arrière de a dans V. Pour un calcul effectif, la division euclidienne

$$\mathrm{L} = \mathrm{Q}(a^{-1}x\partial_x + 1) - (c-a-1)ax^{-1} \qquad \text{où} \qquad \mathrm{Q} = a(1-x)\partial_x + (c-a-1)ax^{-1} - ab$$

donne l'opérateur de descente après avoir décalé a dans $(c-a-1)^{-1}a^{-1}x\mathrm{Q}$ par

$$\mathrm{L}_{\downarrow} = \frac{x(1-x)}{a-c}\partial_x - \frac{bx}{a-c} - 1.$$

Notre objectif est maintenant de calculer une relation de contiguïté pour F. Nous avons obtenu $\mathrm{L}_{\uparrow}(a)\cdot\mathrm{F} = \partial_a\cdot\mathrm{F}$, où nous avons noté explicitement la dépendance en a du polynôme tordu $\mathrm{L}_{\uparrow}$. Il s'ensuit la relation

$$\partial_a^i\cdot\mathrm{F} = \mathrm{L}_{\uparrow,i}(a)\cdot\mathrm{F} \qquad \text{où} \qquad \mathrm{L}_{\uparrow,i}(a) = \mathrm{L}_{\uparrow}(a+i-1)\cdots\mathrm{L}_{\uparrow}(a+1)\mathrm{L}_{\uparrow}(a),$$

dans laquelle nous pouvons, comme toujours, remplacer un polynôme agissant sur la fonction F par le reste de la division euclidienne de ce polynôme par L, qui annule F. Ainsi, une relation de contiguïté s'obtient en recherchant une combinaison linéaire à coefficients dans $\mathbb{Q}(a,b,c,x)$ des restes modulo L à droite des $\mathrm{L}_{\uparrow,i}(a)$ pour $i = 0,1,2$.

Exercice 30.5 Terminer ce calcul pour retrouver :

$$(a+1)(1-x)\mathrm{F}(a+2,b;c;x) + \Big(c - xb + (a+1)(x-2)\Big)\mathrm{F}(a+1,b;c;x) + (a-c+1)\mathrm{F}(a,b;c;x) = 0.$$

■

Extension aux séries partiellement hypergéométriques

Nous considérons maintenant des sommes

$$f_n(x) = \sum_{k\geq 0} u_{n,k}x^k$$

dans lesquelles u n'est hypergéométrique qu'en n, mais est seulement polynomialement récurrente en k, et satisfait à la relation

$$a_p(n,k)u_{n,k+p} + \cdots + a_0(n,k)u_{n,k} = 0.$$

Comme dans le cas doublement hypergéométrique, cette relation fournit une relation purement différentielle sur f. Comme précédemment, aussi, la relation de récurrence du premier ordre en n sur u donne une expression de $f_{n+1}(x)$ comme combinaison linéaire de dérivées. On procède donc comme dans la section précédente pour calculer opérateurs de montée, de descente, et relations de contiguïté.

Exercice 30.6 — Assez calculatoire. Calculer l'opérateur de montée annoncé dans l'introduction de cette section. ■

Notes

La structure d'anneau de polynômes tordus a été introduite et étudiée par Ore dans les années 1930 [Ore33], avec le dessein explicite de traiter des systèmes fonctionnels linéaires [Ore31]. Dans ces travaux, les polynômes tordus sont simplement appelés « non commutatifs ». Dans la littérature moderne, on parle souvent de « polynômes » ou d'« opérateurs de Ore », et encore d'« opérateurs pseudo-linéaires » [BP96].

Notre terminologie de « polynôme tordu » est la traduction de l'anglais « *skew polynomial* », où « *skew* » signifie « de biais », « oblique ». Certains auteurs ont proposé la traduction « polynôme gauche », où « gauche » a le sens de « voilé », par opposition à « plan ». Mais nous avons préféré éviter toute confusion avec des notions algébriques de multiple, module, fraction, etc, pour lesquelles « gauche » a le sens opposé de « droite ». De plus, la littérature note généralement les anneaux de polynômes tordus avec des crochets, par $A[\partial;\sigma,\delta]$, mais les chevrons nous semblent mieux évoquer que l'anneau en question est une algèbre sur A donnée par le générateur ∂ et des relations induites par σ et δ, et n'est en général pas commutatif.

L'interprétation de la Section 30.2 du lien par morphisme entre série et suite de coefficients est implicite dans les travaux sur la théorie des D-modules (voir l'article de Cartier [Car92] pour une exposition simple). Elle a été utilisée de façon explicite pour le cadre binomial par Bostan, Chyzak, Cluzeau et Salvy [Bos+06a].

Les algorithmes de recherche de solutions rationnelles interviennent naturellement comme base des algorithmes de factorisation [BP94b ; Bro92], dont est tiré l'Exercice 30.4.

Les questions de factorisation d'opérateurs linéaires et de résolution mènent à une théorie de Galois : la théorie de Galois différentielle [PS03 ; Sin07] comme celle à différence [PS97] étudient les applications linéaires qui échangent les solutions d'une équations fonctionnelle linéaire. Un succès de la théorie différentielle est l'étude des solutions dites « liouviliennes » d'une équation linéaire différentielle, données explicitement en termes de compositions d'opérations élémentaires (exponentiations, prises de logarithmes, extensions algébriques). La théorie fournit aussi des informations sur les développements asymptotiques des solutions, notamment en présence de singularités faisant intervenir des exponentielles.

Le résultant différentiel de deux équations différentielles a été défini par Berkovich et Tsirulik en 1986 [BT86]. Chardin [Cha91] a défini les sous-résultants de deux équations différentielles. Li [Li00 ; Li98] a étendu la définition aux polynômes de Ore. On trouve, dans les travaux de Grigoriev [Gri90] d'une part et de Li et Nemes [LN97] d'autre part, des algorithmes efficaces pour le calcul du pgcdd de deux opérateurs différentiels ; ces algorithmes ont été étendus aux sous-résultants par Li à la fin des années 1990 [Li00 ; Li98]. En 1991, Chardin a donné une expression du résultant de deux équations différentielles en termes de leurs solutions [Cha91]. Cela a été ensuite généralisé au cadre des polynômes de Ore par Hong [Hon01a ; Hon01b].

La question du calcul algorithmique de relations de contiguïté a été abordée d'abord dans le cas purement hypergéométrique par Takayama [Tak89], puis dans le cas partiellement hypergéométrique par Chyzak et Salvy [CS98].

Bibliographie

Bos+06a Bostan, A., F. Chyzak, T. Cluzeau et B. Salvy (2006). « Low complexity algorithms for linear recurrences ». In : *ISSAC'06 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 31–38.

BP94b Bronstein, M. et M. Petkovšek (1994). « On Ore rings, linear operators and factorisation ». In : *Programmirovanie*, vol. 1. Also available as Research Report 200, Informatik, ETH Zürich, p. 27–44.

BP96 Bronstein, Manuel et Marko Petkovšek (1996). « An introduction to pseudo-linear algebra ». In : *Theoretical Computer Science*, vol. 157, p. 3–33.

Bro92 Bronstein, Manuel (1992). « Linear ordinary differential equations : breaking through the order 2 barrier ». In : *ISSAC'92 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 42–48.

BT86 Berkovich, L. M. et V. G. Tsirulik (1986). « Differential resultants and some of their applications ». In : *Differentsial'nye Uravneniya*, vol. 22, n°5. English translation in : Differential Equations, Plenum Publ. Corp., Vol. 22, no. 5, pp. 530–536. NY Plenum 1986, p. 750–757, 914.

Car92 Cartier, Pierre (1992). « Démonstration "automatique" d'identités et fonctions hypergéométriques (d'après D. Zeilberger) ». In : *Astérisque*, vol. 1991/92, n°206. Séminaire Bourbaki, Exp. No. 746, 3, 41–91.

Cha91 Chardin, Marc (1991). « Differential resultants and subresultants ». In : *Fundamentals of computation theory*. Vol. 529. Lecture Notes in Computer Science. Gosen : Springer-Verlag, p. 180–189.

CS98 Chyzak, Frédéric et Bruno Salvy (1998). « Non-commutative elimination in Ore algebras proves multivariate holonomic identities ». In : *Journal of Symbolic Computation*, vol. 26, n°2, p. 187–227.

Gri90 Grigoriev, D. Yu. (1990). « Complexity of factoring and calculating the GCD of linear ordinary differential operators ». In : *Journal of Symbolic Computation*, vol. 10, n°1, p. 7–37.

Hon01a Hong, Hoon (2001). « Ore principal subresultant coefficients in solutions ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 11, n°3, p. 227–237.

Hon01b — (2001). « Ore subresultant coefficients in solutions ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 12, n°5, p. 421–428.

Li00 Li, Ziming (2000). « Greatest common right divisors, least common left multiples, and subresultants of Ore polynomials ». In : *Mathematics mechanization and applications*. San Diego, CA : Academic Press, p. 297–324.

Li98 — (1998). « A subresultant theory for Ore polynomials with applications ». In : *ISSAC'98 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 132–139.

LN97 Li, Z. et I. Nemes (1997). « A modular algorithm for computing greatest common right divisors of Ore polynomials ». In : *ISSAC'97 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 282–289.

Ore31 Ore, Oystein (1931). « Linear equations in non-commutative fields ». In : *Annals of Mathematics*, vol. 32, p. 463–477.

Ore33 Ore, Oystein (1933). « Theory of non-commutative polynomials ». In : *Annals of Mathematics*, vol. 34, n°3, p. 480–508.

PS03 Put, Marius van der et Michael F. Singer (2003). *Galois theory of linear differential equations*. Vol. 328. Grundlehren der Mathematischen Wissenschaften. Springer-Verlag.

PS97 — (1997). *Galois theory of difference equations*. Vol. 1666. Lecture Notes in Mathematics. Springer-Verlag.

Sin07 Singer, M. F. (2007). « Introduction to the Galois theory of linear differential equations ». url : https://arxiv.org/abs/0712.4124.

Tak89 Takayama, Nobuki (1989). « Gröbner basis and the problem of contiguous relations ». In : *Japan Journal of Applied Mathematics*, vol. 6, n°1, p. 147–160.

31. Algèbres de Ore et fonctions spéciales de plusieurs variables

Résumé

Les polynômes tordus univariés du Chapitre 30 se généralisent à plusieurs indéterminées pour bien modéliser des opérateurs linéaires agissant sur des fonctions et suites à plusieurs variables ou indices. Cette représentation polynomiale ouvre la voie à une théorie algorithmique de leurs idéaux à gauche, qui sont vus comme idéaux annulateurs de fonctions et suites, maintenant représentés commes solutions implicites de systèmes fonctionnels linéaires. Une extension non commutative de l'algorithmique des bases de Gröbner fournit les briques de base pour calculer avec cette nouvelle représentation : les notions de fonctions différentiellement finies (univariées) et de suites polynomialement récurrentes (à un indice) se fondent dans la généralité des fonctions ∂-finies.

31.1 Algèbres de Ore rationnelles

Une généralisation à plusieurs dérivations et décalages de la notion d'anneau de polynômes tordus du chapitre précédent est donnée par la définition qui suit.

Définition 31.1 — Algèbre de Ore. Étant donnés
- un corps (commutatif) $k(x) = k(x_1, \ldots, x_r)$ de fractions rationnelles,
- r endomorphismes σ_i de k-algèbre de $k(x)$, injectifs et commutant deux à deux,
- pour chaque i une σ-dérivation δ_i relative à σ_i, c'est-à-dire pour chaque i un endomorphisme linéaire pour lequel $\delta_i(ab) = \sigma_i(a)\delta_i(b) + \delta_i(a)b$ dès que a et b sont dans $k(x)$, toutes ces σ-dérivations commutant deux à deux et δ_i commutant avec σ_j chaque fois que $i \neq j$,

— r indéterminées ∂_i,
l'algèbre de Ore (rationnelle) notée $k(x_1,\dots,x_r)\langle\partial_1,\dots,\partial_r;\sigma_1,\dots,\sigma_r,\delta_1,\dots,\delta_r\rangle$ ou plus simplement $k(x)\langle\partial;\sigma,\delta\rangle$ est la $k(x)$-algèbre associative engendrée par les ∂_i modulo les relations

$$\partial_i a = \sigma_i(a)\partial_i + \delta_i(a), \qquad \partial_i\partial_j = \partial_j\partial_i,$$

quand a est dans $k(x)$. On note plus simplement $k(x_1,\dots,x_r)\langle\partial_1,\dots,\partial_r;\sigma_1,\dots,\sigma_r\rangle$ ou encore $k(x)\langle\partial;\sigma\rangle$ le cas où tous les δ_i sont nuls.

Exemple 31.1 En notant x l'opérateur de multiplication par x et n celui de multiplication par n, on vérifie l'existence d'une algèbre de Ore

$$\mathrm{A} = \mathbb{C}(n,x)\langle\partial_n,\partial_x;\mathrm{S}_n,\mathrm{id},0,\mathrm{D}_x\rangle$$

avec $\mathrm{S}_n(n) = n+1$, $\mathrm{S}_n(x) = x$, $\mathrm{D}_x(n) = 0$, $\mathrm{D}_x(x) = 1$, et plus généralement $\mathrm{S}_n(a) = a(n+1,x)$ et $\mathrm{D}_x(a) = da/dx$ quand $a = a(n,x) \in \mathbb{C}(n,x)$.

31.2 Idéal annulateur et module quotient

Les éléments des algèbres de Ore représentent des opérateurs linéaires, différentiels, de récurrence, ou autres, et agissent donc sur des fonctions, suites, suites de fonctions, etc. Pour montrer que ces objets sont une bonne représentation polynomiale des opérateurs linéaires, l'exemple de la famille des polynômes orthogonaux de Laguerre va nous servir pour toute la suite du chapitre.

Pour chaque paramètre strictement positif α, l'intégrale

$$(f \mid g) = \int_0^\infty f(x)g(x)x^\alpha e^{-x}\,dx$$

définit un produit scalaire sur les fonctions polynomiales réelles. Par orthogonalisation de Gram–Schmidt (voir la Section 20.2), on déduit l'existence de bases orthogonales échelonnées en degré. On a par exemple la base des polynômes orthogonaux de Laguerre, donnée par

$$\mathrm{L}_n^{(\alpha)}(x) = \frac{1}{n!}x^{-\alpha}e^x\left(\frac{d}{dx}\right)^n(e^{-x}x^{n+\alpha}) = \frac{1}{n!}\sum_{k=0}^{n}(-1)^k\binom{n}{k}(\alpha+k+1)\cdots(\alpha+n)x^k.$$

On vérifie que ces polynômes vérifient les relations (linéaires)

$$\begin{aligned}(n+2)\mathrm{L}_{n+2}^{(\alpha)} - (2n+\alpha+3-x)\mathrm{L}_{n+1}^{(\alpha)} + (n+\alpha+1)\mathrm{L}_n^{(\alpha)} &= 0,\\ x\mathrm{L}_n^{(\alpha)\prime} - (n+1)\mathrm{L}_{n+1}^{(\alpha)} + (n+\alpha+1-x)\mathrm{L}_n^{(\alpha)} &= 0,\\ x\mathrm{L}_n^{(\alpha)\prime\prime} + (\alpha+1-x)\mathrm{L}_n^{(\alpha)\prime} + n\mathrm{L}_n^{(\alpha)} &= 0,\end{aligned}$$

avec les conditions initiales $\mathrm{L}_0^{(\alpha)} = 1$ et $\mathrm{L}_1^{(\alpha)} = \alpha + 1 - x$. Dans l'algèbre de Ore A introduite par l'Exemple 31.1 ci-dessus, ces équations se recodent en les polynômes tordus suivant, qui annulent la suite de fonctions polynomiales $\mathrm{L}^{(\alpha)}$:

$$\begin{aligned} p_1 &= (n+2)\partial_n^2 - (2n+\alpha+3-x)\partial_n + (n+\alpha+1), \\ p_2 &= x\partial_x - (n+1)\partial_n + (n+\alpha+1-x), \\ p_3 &= x\partial_x^2 + (\alpha+1-x)\partial_x + n. \end{aligned}$$

Ces trois polynômes engendrent un idéal à gauche dans A (voir la définition en Section 30.1) et cet idéal est appelé l'*idéal annulateur* de $\mathrm{L}^{(\alpha)}$ dans A. Cette stabilité reflète le fait que l'addition terme à terme de deux relations linéaires vérifiées par $\mathrm{L}^{(\alpha)}$ est une nouvelle relation linéaire vérifiée par $\mathrm{L}^{(\alpha)}$, de même qu'en appliquant un opérateur linéaire sur une relation linéaire vérifiée par $\mathrm{L}^{(\alpha)}$, on retrouve une relation linéaire vérifiée par $\mathrm{L}^{(\alpha)}$.

Exemple 31.2 En partant de la troisième équation donnée pour caractériser les polynômes de Laguerre, un décalage avant de n et une dérivation par rapport à x donnent respectivement

$$x\mathrm{L}_{n+1}^{(\alpha)\,\prime\prime} + (\alpha+1-x)\mathrm{L}_{n+1}^{(\alpha)\,\prime} + (n+1)\mathrm{L}_{n+1}^{(\alpha)} = 0$$

et

$$x\mathrm{L}_n^{(\alpha)\,\prime\prime\prime} + (\alpha+2-x)\mathrm{L}_n^{(\alpha)\,\prime\prime} + (n-1)\mathrm{L}_n^{(\alpha)\,\prime} = 0.$$

L'addition de ces deux équations donne l'équation fonctionnelle linéaire

$$x\mathrm{L}_n^{(\alpha)\,\prime\prime\prime} + (\alpha+2-x)\mathrm{L}_n^{(\alpha)\,\prime\prime} + (n-1)\mathrm{L}_n^{(\alpha)\,\prime} + x\mathrm{L}_{n+1}^{(\alpha)\,\prime\prime} + (\alpha+1-x)\mathrm{L}_{n+1}^{(\alpha)\,\prime} + (n+1)\mathrm{L}_{n+1}^{(\alpha)} = 0,$$

laquelle correspond au polynôme tordu

$$(\partial_x + \partial_n)p_3 = x\partial^3 + (\alpha+2-x)\partial_x^2 + (n-1)\partial_x + x\partial_x^2\partial_n + (\alpha+1-x)\partial_x\partial_n + (n+1)\partial_n.$$

Dans le cas commutatif, le quotient d'une algèbre A de polynômes par l'un de ses idéaux (bilatères) I reste munie d'un produit canonique et est donc une algèbre. Cette propriété n'est plus réalisée dans le cas d'une algèbre de Ore $\mathrm{A} = k(x)\langle\partial;\sigma,\delta\rangle$ et d'un idéal à gauche I. Néanmoins, le quotient A/I conserve une addition canonique, ainsi que la stabilité par multiplication à gauche par tout élément de A, ce qui fait de ce quotient un module à gauche sur A et en particulier un espace vectoriel sur $k(x)$.

Dans le cas commutatif, un cadre particulier important est celui d'un quotient de dimension finie comme espace vectoriel, car il représente une famille finie de points solutions (voir la Section 25.2). Le cas d'un quotient d'une algèbre de Ore qui est un espace vectoriel sur $k(x)$ de dimension finie est lui-aussi important ; dans l'interprétation en opérateurs linéaires, il correspond en règle générale à un espace vectoriel de solutions de dimension finie sur k.

Dans la Section 31.4, nous détaillerons ce lien dans le cas d'opérateurs différentiels ;

d'autres cadres fournissent le même genre de résultats. Nous irons plus loin sur le sujet dans la Section 31.5 sur les fonctions ∂-finies.

Toute autre famille de polynômes orthogonaux classiques se traiterait de la même manière et aurait pu servir de support à ce chapitre. La même nature de système linéaire avec une dérivation sur une variable et un décalage sur une autre permet de traiter de la même façon nombre de familles de fonctions spéciales paramétrées, telles les fonctions de Bessel, de Hankel, etc.

31.3 Bases de Gröbner pour les idéaux à gauche

La propriété essentielle qui fait fonctionner toute la théorie des bases de Gröbner et l'algorithme de Buchberger dans le cadre de polynômes commutatifs est que le monôme de tête d'un produit de polynômes est le produit des monômes de tête des termes du produit. Cette propriété reste vérifiée sur des polynômes non commutatifs sujets aux relations de définition des algèbres de Ore rationnelles, dès lors qu'on considère des ordres monomiaux sur les ∂_i. En refaisant la théorie en s'efforçant de faire toutes les combinaisons linéaires avec des facteurs à gauche, on obtient le résultat suivant (*cf.* le Chapitre 23 et la Section 24.3 sur les bases de Gröbner classiques) :

Théorème 31.1 Soit A une algèbre de Ore rationnelle.
i. Tout idéal à gauche I de A admet pour chaque ordre monomial (admissible) sur les ∂_i une unique base de Gröbner minimale réduite G, au sens où l'une quelconque des propriétés équivalentes suivantes est vérifiée :

1. la partie stable du monoïde des monômes en les ∂_i engendrée par les monômes de tête des éléments de G est égale à celle engendrée par ceux de I ;
2. tout f non nul de I est réductible par G ;
3. pour tout f dans A, il existe un unique r dans A dont aucun monôme ne soit divisible par un monôme de tête d'un élément de G et tel que $f - r$ soit dans l'idéal I ;
4. pour tout f dans I, le reste de la division (à droite) de f par G est nul.

ii. Soit $\mathrm{P} = \{p_k\}_{1 \leq k \leq r}$ un système de générateurs non nuls d'un idéal à gauche I de A. Tous les S-polynômes $\mathrm{Spoly}(p_i, p_j)$ (définis par des combinaisons linéaires à gauche) se réduisent à 0 par P si et seulement si P est une base de Gröbner de l'idéal.
iii. Une variante de l'algorithme de Buchberger termine et renvoie une base de Gröbner de tout idéal à gauche I de A.

Démonstration. Nous nous cantonnons ici à montrer l'équivalence des quatre propriétés du point *i*, par une preuve cyclique. Supposons 1. et soit f non nul de I : le monôme de tête f est donc multiple d'un monôme de tête d'un élément de G, et f est ainsi réductible par G. Supposons 2., soit f dans A, et soit r le reste de la division multivariée non commutative de f par G. Alors r remplit les conditions d'existence demandées par le point 3. ; si r' remplit les mêmes conditions, alors $r - r' = (f - r') - (f - r)$ est dans I. Si cette différence est non nulle, alors par l'hypothèse 2. elle est réductible par G : son monôme de tête, qui doit être un monôme de r ou de r, est divisible par un monôme de tête de G, ce qui est une contradiction. Donc $r = r'$, prouvant

ainsi l'unicité. Supposons 3. et soit f dans I : l'hypothèse 3. s'applique pour $r = 0$, qui est donc le reste nécessaire de la division multivariée non commutative de f par G. Supposons enfin 4. et considérons un monôme m élément de la partie stable engendrée par les monômes de tête de I. Cet m est donc multiple (à gauche) d'un monôme de tête d'un élément de G. La partie stable engendrée par les monômes de tête des éléments de G inclut donc celle engendrée par ceux de I ; l'autre inclusion étant évidente vu que les éléments de G sont dans I, il y a égalité. ■

Une spécificité du cas non commutatif réside dans le calcul des S-polynômes. Considérons le cas commutatif et faisons l'hypothèse que les calculs, pour des raisons d'efficacité, écrivent tous les polynômes en chassant les dénominateurs. (Ceci suppose donc que le corps des coefficients peut être vu comme l'anneau des fractions d'un certain anneau, en règle générale $\mathbb{Z}[\alpha,\dots]$ pour des paramètres α, ...) Le S-polynôme de deux polynômes non nuls f_1 et f_2 se définit alors par

$$\mathrm{Spoly}(f_1,f_2) = \frac{c_2}{\mathrm{pgcd}(c_1,c_2)}\frac{m_2}{\mathrm{pgcd}(m_1,m_2)}f_1 - \frac{c_1}{\mathrm{pgcd}(c_1,c_2)}\frac{m_1}{\mathrm{pgcd}(m_1,m_2)}f_2,$$

où c_i dénote le coefficient de tête de f_i et m_i son monôme de tête, pour $i = 1,2$. Cette formule doit être adaptée dans le cas de polynômes tordus : c_1 et c_2 dénotent maintenant respectivement le coefficient de tête de

$$\frac{m_2}{\mathrm{pgcd}(m_1,m_2)}f_1 \qquad \text{et} \qquad \frac{m_1}{\mathrm{pgcd}(m_1,m_2)}f_2,$$

qui peuvent dépendre des σ_j, mais où les notions divisions et pgcd sont vues comme si m_1 et m_2 étaient dans un monde commutatif. Plutôt que de poursuivre la théorie dans le détail, montrons le calcul sur un exemple.

Exemple 31.3 En repartant des polynômes p_1 et p_2 qui annulent la suite des polynômes orthogonaux de Laguerre, montrons que le polynôme p_3 s'obtient par élimination de ∂_n par un calcul pour l'ordre lexicographique induit par $\partial_n > \partial_x$. Pour cet ordre, le terme de tête de p_1 est $(n+2)\partial_n^2$, celui de p_2 est $-(n+1)\partial_n$. On calcule donc d'abord le S-polynôme de p_1 et de p_2, sous la forme

$$\mathrm{Spoly}(p_1,p_2) = p_1 + \partial_n p_2 = x\partial_x\partial_n - (n+1)\partial_n + (n+\alpha+1).$$

Remarquons que ce S-polynôme n'est pas $(n+1)p_1 + (n+2)\partial_n p_2$, lequel aurait $(n+2)\partial_n^2$ comme terme de tête, et non pas $\partial_x\partial_n$ pour monôme de tête. Il est réductible par p_2 ; après multiplication par $(n+1)$ et ajout de $x\partial_x p_2$, on obtient

$$-(n+1)^2\partial_n + x^2\partial_x^2 + (n+\alpha+2-x)x\partial_x + (n+1)(n+\alpha+1) - x.$$

Ce polynôme a ∂_n pour monôme de tête et est réductible par p_2 ; après retranchement de $(n+1)p_2$, on aboutit à

$$x^2\partial_x^2 + (\alpha+1-x)x\partial_x + nx,$$

qui n'est autre que xp_3. En poursuivant, on montre que les S-polynômes de p_1 et p_2

avec p_3 se réduisent à 0 ; puisque le monôme de tête de p_2, ∂_n, divise celui de p_1, ∂_n^2, une base de Gröbner minimale pour l'ordre $\mathrm{lex}(\partial_n, \partial_x)$ est $\{p_2, p_3\}$.

De façon analogue à l'exemple précédent, une base de Gröbner pour l'ordre $\mathrm{lex}(\partial_x, \partial_n)$ de l'idéal engendré par p_2 et p_3 est $\{p_1, p_2\}$. Les bases de Gröbner permettent ainsi de déterminer la redondance du système $\{p_1, p_2, p_3\}$.

Exercice 31.1 Calculer une base de Gröbner pour l'ordre $\mathrm{lex}(\partial_x, \partial_n)$ de l'idéal engendré par p_2 et p_3 et vérifier le point ci-dessus. ■

Les polynômes p_1, p_2, p_3 qui annulent la suite des polynômes orthogonaux de Laguerre sont encore plus contraints qu'il n'y paraît jusqu'à présent : p_2 se déduit en fait de p_1. En effet, ne connaissant que p_1, on peut rechercher le polynôme p_2 sous la forme indéterminée

$$p_2 = \partial_x - u(n,x)\partial_n - v(n,x),$$

pour des fractions rationnelles à déterminer u et v, et faire l'hypothèse heuristique que $\{p_1, p_2\}$ est une base de Gröbner pour l'ordre $\mathrm{lex}(\partial_x, \partial_n)$. (Cette hypothèse heuristique est en fait naturelle dès qu'on sait qu'on a à faire à une famille de polynômes orthogonaux.) La contrainte d'être une base de Gröbner implique que le S-polynôme $\mathrm{Spoly}(p_1, p_2)$ doit se réduire à 0 par $\{p_1, p_2\}$, ce qui impose certaines relations différentielles et à différence sur u et v.

Exercice 31.2 Utiliser la théorie des bases de Gröbner pour donner un système de récurrences linéaires sur u et v qui, après résolution, redonne le polynôme p_2. (Pour la résolution, on se souviendra des conditions initiales $\mathrm{L}_0^{(\alpha)} = 1$ et $\mathrm{L}_1^{(\alpha)} = \alpha + 1 - x$.) ■

31.4 Module quotient et dimension de l'espace des solutions

Séries formelles solutions en un point ordinaire dans le cas différentiel

Considérons une algèbre de Ore

$$\mathrm{A} = \mathbb{C}(x_1, \ldots, x_r)\langle \partial_1, \ldots, \partial_r; \mathrm{id}, \ldots, \mathrm{id}, \mathrm{D}_{x_1}, \ldots, \mathrm{D}_{x_r}\rangle,$$

un idéal à gauche I de cette algèbre, donné par un système différentiel linéaire. Nous voulons décrire les solutions séries annulées par tous les éléments de I, où une série est ici un élément de $\mathbb{C}[[x_1, \ldots, x_r]]$, c'est-à-dire une combinaison linéaire formelle éventuellement infinie de monômes à exposants entiers positifs. Dans cet objectif, cette section fournit un analogue en plusieurs variables de la conversion entre équation différentielle décrivant une série différentiellement finie d'une variable et équation de récurrence vérifiée par la suite polynomialement récurrente des coefficients.

Fixons un ordre monomial sur les monômes en les ∂_i, puis, pour cet ordre, une base de Gröbner B de I, donnée par des éléments de A sans fractions, c'est-à-dire avec des coefficients polynomiaux. Cette base de Gröbner B fournit un escalier ; notons S l'ensemble des multi-exposants $s = (s_1, \ldots, s_r)$ des monômes $\partial^s = \partial_1^{s_1} \cdots \partial_r^{s_r}$ sous l'escalier, c'est-à-dire des monômes qui ne sont pas réductibles par B. Le module quotient A/I a alors une base d'espace vectoriel sur $\mathbb{C}(x)$ constituée des $\partial^s + \mathrm{I}$, les classes des ∂^s modulo I, pour s décrivant S. Soit u le polynôme produit des coefficients

de tête des éléments de B et faisons l'hypothèse que u ne s'annule pas pour $x_1 = \cdots = x_r = 0$. Nous affirmons qu'alors, l'idéal I admet un espace vectoriel de solutions séries dans $\mathbb{C}[[x_1, \ldots, x_r]]$ de dimension le cardinal de S, c'est-à-dire la dimension sur $\mathbb{C}(x)$ de A/I vu comme espace vectoriel. On dit dans ce cas que le point $(0, \ldots, 0)$ est régulier pour le système différentiel linéaire définissant l'idéal I.

En effet, pour tout multi-exposant $n = (n_1, \ldots, n_r)$, la réduction du monôme ∂^n par B fournit une combinaison linéaire $\sum_{s\in S} v_{n,s}\partial^s$ congrue à ∂^n modulo I. Ceci généralise la récriture d'un ∂^N faite en Section 30.3. Notons que par construction, les coefficients $v_{n,s}$ sont éléments de $\mathbb{C}[x_1, \ldots, x_r, u^{-1}]$ et ont ainsi une évaluation bien définie en $x_1 = \cdots = x_r = 0$. Par suite, puisque chaque élément de I s'annule sur toute solution série

$$\phi = \sum_{n_1\in\mathbb{N},\ldots,n_r\in\mathbb{N}} c_{n_1,\ldots,n_r} x_1^{n_1} \cdots x_r^{n_r},$$

le monôme ∂^n et la somme $\sum_{s\in S} v_{n,s}\partial^s$ ont la même action sur ϕ :

$$\partial^n \cdot \phi = \sum_{s\in S} v_{n,s}\partial^s \cdot \phi.$$

Une évaluation en $x_1 = \cdots = x_r = 0$ donne la relation

$$n_1! \cdots n_r!\, c_{n_1,\ldots,n_r} = \sum_{s\in S} v_{n,s}(0, \ldots, 0) s_1! \cdots s_r!\, c_{s_1,\ldots,s_r}.$$

Autrement dit, la série ϕ est totalement déterminée par ses quelques premiers coefficients c_s pour $s \in S$, en nombre donné par la dimension de A/I.

L'exemple des polynômes orthogonaux de Laguerre étend déjà légèrement le cadre purement différentiel qui précède. Posons

$$L_n^{(\alpha)}(x) = \sum_{k=0}^{n} \ell_{n,k} x^k.$$

En multipliant chaque p_i pour $i = 1, 2, 3$ par ∂_x^k, il vient

$$\begin{aligned}
\partial_x^k p_1 &= (n+2)\partial_n^2\partial_x^k - (2n+\alpha+3-x)\partial_n\partial_x^k + k\partial_n\partial_x^{k-1} + (n+\alpha+1)\partial_x^k,\\
\partial_x^k p_2 &= x\partial_x^{k+1} - (n+1)\partial_n\partial_x^k + (n+\alpha+k+1-x)\partial_x^k - k\partial_x^{k-1},\\
\partial_x^k p_3 &= x\partial_x^{k+2} + (\alpha+k+1-x)\partial_x^{k+1} + (n-k)\partial_x^k.
\end{aligned}$$

Après application sur $L^{(\alpha)}$, évaluation en $x = 0$ et division par $k!$, on trouve les relations de récurrence sur la famille doublement indexée des $\ell_{n,k}$

$$\begin{aligned}
(n+2)\ell_{n+2,k} - (2n+\alpha+3)\ell_{n+1,k} + \ell_{n+1,k-1} + (n+\alpha+1)\ell_{n,k} &= 0,\\
-(n+1)\ell_{n+1,k} + (n+\alpha+k+1)\ell_{n,k} - \ell_{n,k-1} &= 0,\\
(k+1)(\alpha+k+1)\ell_{n,k+1} + (n-k)\ell_{n,k} &= 0.
\end{aligned}$$

En décalant la dernière vers l'arrière en k puis éliminant $\ell_{n,k-1}$ entre la relation obtenue et la deuxième récurrence ci-dessus, on obtient la récurrence

$$(n+1-k)\ell_{n+1,k} - (n+\alpha+1)\ell_{n,k} = 0.$$

Ce jeu de récurrences permet de calculer tous les $\ell_{n,k}$.

Solutions en séries des systèmes hypergéométriques de Gel'fand, Kapranov et Zelevinsky

Un exemple concret est donné par une généralisation à plusieurs variables des séries hypergéométriques de la Section 29.1. Considérons l'algèbre de Ore A engendrée par quatre indéterminées $\partial_1, \ldots, \partial_4$ sur le corps $\mathbb{C}(x_1,\ldots,x_4)$, chaque ∂_i représentant l'opérateur de dérivation par rapport à x_i, et le système

$$p_1 = \partial_2\partial_3 - \partial_1\partial_4, \qquad p_2 = x_1\partial_1 - x_4\partial_4 + (1-c),$$
$$p_3 = x_2\partial_2 + x_4\partial_4 + a, \qquad p_4 = x_3\partial_3 + x_4\partial_4 + b, \quad (31.1)$$

pour des paramètres complexes a, b et c génériques. L'objectif de l'exemple est de montrer que ce système admet un espace vectoriel de solutions formelles de dimension exactement 2, où par solution formelle nous entendons maintenant une série multivariée de la forme

$$x_1^{a_1}\cdots x_4^{a_4} \sum_{n_1\in\mathbb{Z},\ldots,n_4\in\mathbb{Z}} c_{n_1,\ldots,n_4} x_1^{n_1}\cdots x_4^{n_4},$$

pour des a_i et des coefficients complexes, ou une combinaison linéaire de telles séries. (Il y a bien un espace vectoriel sur $\mathbb{C}$ où vivent ces séries, mais pas de produit sur ces séries. En revanche, toute série peut être multipliée par un polynôme en les x_i et leurs inverses x_i^{-1}, ainsi que dérivée formellement par rapport à chacune des indéterminées, tout en restant dans l'espace vectoriel.)

Soit I l'idéal engendré par le système $\{p_1,\ldots,p_4\}$ et calculons à partir de ce système une base de Gröbner de I pour l'ordre $1\mathrm{ex}(\partial_1,\ldots,\partial_4)$. Les monômes de tête respectifs de p_1 et p_2 sont $\partial_1\partial_4$ et ∂_1. Le S-polynôme de p_1 et de p_2 est donc

$$\mathrm{Spoly}(p_1,p_2) = x_1p_1 + \partial_4 p_2 = x_1\partial_2\partial_3 - x_4\partial_4^2 - c\partial_4,$$

dont le monôme de tête est $\partial_2\partial_3$; il est donc réductible par p_3. Après multiplication par $-x_2$ et ajout de $x_1\partial_3 p_3$, on obtient

$$x_1x_4\partial_3\partial_4 + ax_1\partial_3 + x_2x_4\partial_4^2 + cx_2\partial_4.$$

Ce polynôme a $\partial_3\partial_4$ pour monôme de tête et est donc réductible par p_4. Après multiplication par x_3 et retranchement de $x_1x_4\partial_4 p_4$, on aboutit à

$$ax_1x_3\partial_3 + (x_2x_3 - x_1x_4)x_4\partial_4^2 + \Big(cx_2x_3 - (b+1)x_1x_4\Big)\partial_4,$$

qui est encore réductible par p_4. Après retranchement de ax_1p_4, on a finalement un polynôme qui n'est pas réductible par $\{p_1,\ldots,p_4\}$, à savoir

$$p_5 = (x_2x_3 - x_1x_4)x_4\partial_4^2 + \Big(cx_2x_3 - (a+b+1)x_1x_4\Big)\partial_4 - abx_1.$$

Par ailleurs, les S-polynômes entre les polynômes p_2, p_3 et p_4 pris deux à deux sont tous nuls, comme on le vérifie en observant que les $x_i\partial_i$ commutent deux à deux. En poursuivant les calculs sur les S-polynômes $\mathrm{Spoly}(p_i,p_5)$, on montre que tous ces derniers se réduisent à 0 par $\{p_1,\ldots,p_5\}$. On obtient ainsi qu'une base de Gröbner minimale est $\{p_2,p_3,p_4,p_5\}$, avec les monômes de tête respectifs ∂_1, ∂_2, ∂_3 et ∂_4^2.

Le module quotient A/I a donc une base d'espace vectoriel sur $\mathbb{C}(x_1,\ldots,x_4)$ constituée de $1+\mathrm{I}$ et $\partial_4+\mathrm{I}$, les classes respectives de 1 et ∂_4 modulo I. La structure de module est donnée explicitement par l'action des ∂_i sur ces deux éléments de base.

Exercice 31.3 Donner l'expression explicite de cette action en récrivant chaque $\partial_i \cdot (1 + \mathrm{I})$ et chaque $\partial_i \cdot (\partial_4 + \mathrm{I})$ sur la base $(1 + \mathrm{I}, \partial_4 + \mathrm{I})$. ∎

Revenons sur les solutions séries du système (31.1). Le polynôme p_2 agit sur un monôme par

$$p_2 \cdot (x_1^{\lambda_1} \cdots x_4^{\lambda_4}) = (\lambda_1 - \lambda_4 + 1 - c) x_1^{\lambda_1} \cdots x_4^{\lambda_4}.$$

(Notons la distinction entre le produit dans A noté $p_2 x_1^{\lambda_1} \cdots x_4^{\lambda_4}$ et l'opération de p_2, ici sur une serie h en x, notée $p_2 \cdot h$; on comparera par exemple $\partial_1 x_1^5 = x_1^5 \partial_1 + 5x_1^4$ et $\partial_1 \cdot x_1^5 = 5x_1^4$.) Ainsi, un monôme $x_1^{\lambda_1} \cdots x_4^{\lambda_4}$ ne peut apparaître avec un coefficient non nul dans une série ϕ solution du système (31.1) que si $\lambda_1 - \lambda_4 + 1 - c$ est nul. En poursuivant ce type de raisonnement avec p_3 et p_4, on obtient de même les contraintes $\lambda_2 + \lambda_4 + a = 0$ et $\lambda_3 + \lambda_4 + a = 0$ et on aboutit à ce que les seuls monômes pouvant apparaître avec un coefficient non nul sont de la forme

$$x_1^{\lambda_4 + c - 1} x_2^{-\lambda_4 - a} x_3^{-\lambda_4 - b} x_4^{\lambda_4} = \frac{x_1^{c-1}}{x_2^a x_3^b} \left(\frac{x_1 x_4}{x_2 x_3} \right)^{\lambda_4},$$

et une solution ϕ est nécessairement de la forme

$$\phi = \frac{x_1^{c-1}}{x_2^a x_3^b} f\left(\frac{x_1 x_4}{x_2 x_3} \right),$$

pour une série formelle f en y à exposants entiers relatifs. Reste à exploiter que ϕ est solution de p_1, ou de manière équivalente de p_5, puisque sous la forme ci-dessus ϕ est déjà solution de p_2, p_3 et p_4. Après avoir évalué en $y = x_1 x_4 / x_2 x_3$, on a

$$0 = \frac{x_2^a x_3^b}{x_1^{c-2}} p_5 \cdot \phi = (1 - y) y f''(y) + \big(c - (a + b + 1) y \big) f'(y) - ab f(y).$$

On reconnaît là l'équation hypergéométrique de Gauss, annulée par la série

$$f_1 = {}_2\mathrm{F}_1 \left(\begin{matrix} a, b \\ c \end{matrix} \middle| y \right) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n}{(c)_n} \frac{y^n}{n!}$$

déjà rencontrée en Section 30.6. On vérifie qu'une solution formelle linéairement indépendante avec f_1 est $f_2 = y^{1-c} {}_2\mathrm{F}_1 \left(\begin{matrix} a - c + 1, b - c + 1 \\ 2 - c \end{matrix} \middle| y \right)$. On a ainsi obtenu deux solutions formelles linéairement indépendantes du système (31.1),

$$\phi_i = (x_1^{c-1} / x_2^a x_3^b) f_i(x_1 x_4 / x_2 x_3) \qquad \text{pour } i = 1 \text{ et } i = 2.$$

Pour tout système différentiel représenté par un idéal I de A, un résultat d'analyse, le théorème de Cauchy–Kovalevskaya, affirme l'existence, au voisinage de tout point en dehors d'une certaine variété (algébrique affine) S, d'un $\mathbb{C}$-espace vectoriel de solutions analytiques de dimension celle sur $\mathbb{C}(x)$ de l'espace vectoriel A/I. Or, on

montre que cette variété S est incluse dans le lieu des zéros du produit des coefficients polynomiaux de tête d'une base de Gröbner de I écrite sans fractions.

Dans le cas de notre exemple, la dimension de A/I est 2 et la variété S est incluse dans le lieu des zéros de $x_1 \cdots x_4(x_2x_3 - x_1x_4)$. Hors de ce lieu, y n'est ni nul, ni infini, ni égal à 1, et les fonctions ϕ_1 et ϕ_2 sont donc analytiques puisque, moyennant quelques hypothèses sur les paramètres a, b et c, les deux séries solutions de l'équation de Gauss, f_1 et f_2, représentent des fonctions analytiques sur $\mathbb{C} \setminus \{0, 1\}$. On a donc trouvé un espace de solutions analytiques de dimension 2, et par le théorème de Cauchy–Kovalevskaya, toutes les solutions analytiques du système (31.1) en dehors de sa variété S pour a, b et c génériques.

31.5 Les fonctions ∂-finies et leurs clôtures

Nous poursuivons maintenant avec le cas particulier important des systèmes fonctionnels linéaires correspondant à des modules A/I de dimension finie sur $\mathbb{C}(x)$. L'objectif est ici de montrer que pour une algèbre A donnée, leurs solutions, que nous appellerons « fonctions ∂-finies », forment une algèbre sur $\mathbb{C}$. Nous allons donner un algorithme pour calculer les clôtures correspondantes. Voici tout de suite la définition, déjà motivée par les sections et chapitres précédents sur les fonctions différentiellement finies et les suites polynomialement récurrentes (en particulier, le Chapitre 14).

Définition 31.2 — Fonction ∂-finie. Étant donnée une algèbre de Ore (rationnelle)

$$\mathrm{A} = \mathbb{C}(x_1, \ldots, x_r)\langle \partial_1, \ldots, \partial_r; \sigma_1, \ldots, \sigma_r, \delta_1, \ldots, \delta_r \rangle$$

agissant sur un $\mathbb{C}(x_1, \ldots, x_r)$-espace vectoriel V, un élément f de V est dit ∂-fini lorsque les conditions équivalentes suivantes sont vérifiées :

1. pour chaque i entre 1 et r, il existe un polynôme $\mathrm{P}_i = \mathrm{P}_i(x_1, \ldots, x_r, \partial_i)$ non nul dont l'action annule f ;
2. la famille des $\partial^a \cdot f$, où ∂^a décrit les monômes de A, engendre un espace vectoriel de dimension finie sur $\mathbb{C}(x)$;
3. le module quotient A/I où I note l'idéal annulateur de f pour l'action de A est un espace vectoriel de dimension finie sur $\mathbb{C}(x)$.

Démonstration. Vérifions l'équivalence entre les trois points de la définition ci-dessus. Faisons d'abord l'hypothèse de la première condition, en notant d_i le degré de P_i en ∂_i. Alors, tout ∂^a a un reste par division par les P_i qui s'exprime comme combinaison linéaire des $\partial_1^{j_1} \cdots \partial_r^{j_r}$ pour $0 \leq j_1 < d_1, \ldots, 0 \leq j_r < d_r$. Ainsi, la famille des $\partial^a \cdot f$ reste dans l'espace vectoriel engendré par les $\partial_1^{j_1} \cdots \partial_r^{j_r} \cdot f$ pour $0 \leq j_1 < d_1, \ldots, 0 \leq j_r < d_r$, et la deuxième condition est prouvée. De manière réciproque, pour chaque i, la famille des $\partial_i^j \cdot f$ est une sous-famille de celle des $\partial^a \cdot f$; quand cette dernière engendre un espace vectoriel de dimension finie, la première est une famille liée, ce qui fournit un polynôme non nul P_i. Enfin, l'application linéaire sur A qui à P de A associe $\mathrm{P} \cdot f$ dans l'espace vectoriel V engendré par les $\partial^a \cdot f$ est surjective, de noyau I. Les A-modules

V et A/I sont donc isomorphes, et ils le sont donc aussi en tant qu'espaces vectoriels sur $\mathbb{C}(x)$. Ceci prouve l'équivalence entre les deux dernières conditions. ■

Par commodité, nous appellerons « fonctions » les éléments de l'espace vectoriel V, ce quand bien même il ne s'agirait pas de fonctions au sens de l'analyse, mais afin d'éviter la terminologie plus lourde et moins imagée d'« éléments ∂-finis d'un module sur A » et celle plus répétitive de « fonctions, suites ou suites de fonctions ».

Méthode du vecteur cyclique et généralisation à plusieurs variables

Les algorithmes envisagés pour les clôtures des fonctions ∂-finies s'appuient sur le calcul de vecteur cyclique, dont nous donnons d'abord la définition et en décrivons le calcul dans le cadre classique univarié commutatif. Étant donné un espace vectoriel V sur un corps k, sur lequel on suppose donnée une action de $k[\mathrm{X}]$, un vecteur $v \in \mathrm{V}$ est dit *cyclique* si la famille $\{\mathrm{X}^i \cdot v\}$ engendre V comme k-espace vectoriel. Alors, v engendre V comme $k[\mathrm{X}]$-module. Pour calculer, on suppose que V est de dimension finie d et que l'action de X est donnée sur une base $\mathrm{B} = (b_1, \ldots, b_d)$ de V par une matrice M telle que $\mathrm{X} \cdot w = (a_1, \ldots, a_d)\mathrm{M}\,{}^t\mathrm{B}$ pour tout vecteur $w = a_1 b_1 + \cdots + a_d b_d = (a_1, \ldots, a_d)\,{}^t\mathrm{B}$. Pour tester si v est cyclique et le cas échéant rendre explicite la structure de module de V, on range dans une matrice les lignes $(a_1, \ldots, a_d)\mathrm{M}^i$ pour $0 \leq i \leq m$ avec m à déterminer et on cherche, par exemple par l'algorithme de Gauss, une dépendance linéaire entre les lignes de la matrice obtenue. En procédant successivement avec $m = 0, 1, 2, \ldots$, la première dépendance linéaire fournit le polynôme minimal de v sous l'action de X sur V ; son degré m vérifie $m \leq d$.

Ce calcul s'étend au cadre commutatif de l'action d'une algèbre

$$k[\mathrm{X}] = k[\mathrm{X}_1, \ldots, \mathrm{X}_r]$$

de polynômes en plusieurs indéterminées. Chaque X_i correspond alors à une matrice M_i et la commutativité des X_i dans $k[\mathrm{X}]$ induit la commutativité entre les matrices M_i. Au lieu d'itérer sur les monômes X^i par ordre croissant de i, on itère maintenant sur les monômes $\mathrm{X}^\lambda = \mathrm{X}_1^{\lambda_1} \cdots \mathrm{X}_r^{\lambda_r}$ selon tout ordre qui assure qu'un monôme n'est considéré qu'après tous ses diviseurs. Soit $\lambda(0)$, $\lambda(1)$, etc, l'ordre dans lequel les multi-exposants des monômes sont énumérés. À chaque étape, on recherche une dépendance linéaire entre des vecteurs

$$\mathrm{X}^{\lambda(0)} \cdot v, \ldots, \mathrm{X}^{\lambda(m)} \cdot v,$$

obtenue en recherchant une dépendance linéaire entre les vecteurs de coefficients

$$(a_1, \ldots, a_d)\mathrm{M}^{\lambda(0)}, \ldots, (a_1, \ldots, a_d)\mathrm{M}^{\lambda(m)}, \tag{31.2}$$

où maintenant $\mathrm{M}^{\lambda(j)}$ est le produit $\mathrm{M}_1^{\lambda(j)_1} \cdots \mathrm{M}_d^{\lambda(j)_d}$. En cas d'échec, on conserve ces $m+1$ vecteurs et on reprend la recherche après avoir ajouté le nouveau vecteur $\mathrm{X}^{\lambda(m+1)} \cdot v$; en cas de succès, on retire le dernier vecteur introduit, $\mathrm{X}^{\lambda(m)} \cdot v$, on évite par la suite tous les multiples du monôme $\mathrm{X}^{\lambda(m)}$, et on introduit le nouveau vecteur $\mathrm{X}^{\lambda(m+1)} \cdot v$ pour reprendre la recherche sur la famille

$$\mathrm{X}^{\lambda(0)} \cdot v, \ldots, \mathrm{X}^{\lambda(m-1)} \cdot v, \mathrm{X}^{\lambda(m+1)} \cdot v.$$

Ce calcul termine si et seulement si le quotient $k[X]/I$, vu comme k-espace vectoriel, est de dimension finie. Chaque dépendance linéaire calculée fournit un polynôme P tel que $P(X_1,\dots,X_r)\cdot v = 0$. Lorque l'itération sur les monômes X^λ suit l'ordre croissant selon un ordre monomial (admissible), l'ensemble des polynômes annulateurs obtenus constitue une base de Gröbner de I pour l'ordre choisi, car l'itération maintient une famille de X^λ qui sont tous strictement sous l'escalier, sauf au plus un élément qui est alors sur l'escalier. Par construction, les P obtenus ont cet élément sur un coin (rentrant vers l'origine) de l'escalier.

Exemple 31.4 Soient V le $\mathbb{Q}$-espace vectoriel de dimension 4 et de base $(b_1,\dots,b_4)$, et $v = (0,1,1,0)$ un élément particulier de V. Introduisons l'action de $\mathbb{Q}[X,Y]$ sur V donnée par $X\cdot w = aM_X$ et $Y\cdot w = aM_Y$ pour

$$M_X = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 2 & 0 & 2 & 2 \\ -1/2 & 1/2 & 0 & -3/2 \\ 8 & -5 & -2 & 0 \end{pmatrix}, \qquad M_Y = \begin{pmatrix} 0 & 0 & 1 & 0 \\ -1/2 & 1/2 & 0 & -3/2 \\ 0 & 0 & 0 & 1 \\ -11/2 & 7/2 & 1 & -1/2 \end{pmatrix},$$

quand $w = a_1b_1 + \cdots + a_4b_4$. Un calcul direct donne $M_XM_Y = M_YM_X$.

Choisissons d'abord l'ordre lexicographique pour lequel $Y < X$, si bien que les monômes à considérer sont d'abord 1, Y, Y^2, etc, puis X, XY, XY^2, etc. Les familles successives (1), (1, Y), $(1, Y, Y^2)$ et $(1, Y, Y^2, Y^3)$ donnent toutes des familles (31.2) linéairement indépendantes. La famille $(1, Y, Y^2, Y^3, Y^4)$ donne ensuite la relation

$$(-1 + 6Y + 4Y^2 + Y^4)\cdot v = 0.$$

On remplace donc Y^4 par X pour considérer la famille $(1, Y, Y^2, Y^3, X)$ qui donne la nouvelle relation

$$(-11 + 2Y - Y^2 - 2Y^3 + 7X)\cdot v = 0.$$

Choisissons ensuite l'ordre gradué pour lequel $Y < X$, si bien que les monômes à considérer sont d'abord 1, Y, X, Y^2, XY, X^2, Y^3, XY^2, etc. Chaque famille préfixe de $(1, Y, X, Y^2)$ ne donne aucune relation de dépendance. La famille $(1, Y, X, Y^2, XY)$ donne ensuite la relation

$$(1 - X + 3Y^2 + 2XY)\cdot v = 0,$$

puis la famille $(1, Y, X, Y^2, X^2)$ la relation

$$(-2 - 2Y - 2Y^2 + X^2)\cdot v = 0,$$

et enfin la famille $(1, Y, X, Y^2, Y^3)$ la relation

$$(11 - 2Y - 7X + Y^2 + 2Y^3)\cdot v = 0.$$

Algorithmes de clôture des fonctions ∂-finies

Le procédé du vecteur cyclique s'étend au cas de l'action d'une algèbre de Ore (rationnelle) en présence de fonctions ∂-finies. Soit une algèbre de Ore

$$\mathrm{A} = \mathbb{C}(x)\langle \partial_1, \ldots, \partial_r; \sigma_1, \ldots, \sigma_r, \delta_1, \ldots, \delta_r \rangle,$$

pour laquelle nous faisons l'hypothèse que les σ_ℓ sont surjectifs. L'espace vectoriel utilisé est un module du type A/I, vu comme espace vectoriel sur $\mathbb{C}(x)$, ou même un module obtenu à partir de quelques constructions de base sur des modules de la forme A/I, comme on va le voir sur l'exemple plus bas. L'espace V étant d'une certaine dimension finie d et une base $\mathrm{B} = (b_1, \ldots, b_d)$ de V étant fixée, l'action de chaque ∂_i sur un vecteur $v = a_1 b_1 + \cdots + a_d b_d = a\,{}^t\mathrm{B}$ de coordonnées $a = (a_1, \ldots, a_d)$ est donnée par une matrice M_i indépendante de v sous la forme

$$\partial_i \cdot v = \partial_i \cdot (a\,{}^t\mathrm{B}) = \big(\sigma_i(a)\mathrm{M}_i + \delta_i(a)\big)\,{}^t\mathrm{B}, \tag{31.3}$$

en adoptant une notation selon laquelle les σ_i et δ_i agissent distributivement sur les coefficients de vecteurs ou de matrices. Pour le choix particulier $v = b_\ell$, c'est-à-dire $a_i = 1$ quand $i = \ell$ et $a_i = 0$ sinon, on observe que la ℓ-ième ligne de M_i n'est autre que le vecteur ligne des composantes de $\partial_i \cdot b_\ell$ sur la base B, de sorte que $\partial_i \cdot {}^t\mathrm{B} = \mathrm{M}_i\,{}^t\mathrm{B}$ (avec encore une notation où ∂_i agit distributivement sur les coefficients du vecteur).

En faisant maintenant agir ∂_j sur (31.3) et en posant $a' = \sigma_i(a)\mathrm{M}_i + \delta_i(a)$, on a

$$\begin{aligned}
\partial_j\partial_i \cdot v = \partial_j \cdot (a'\,{}^t\mathrm{B}) &= \big(\sigma_j(a')\mathrm{M}_j + \delta_j(a')\big)\,{}^t\mathrm{B} \\
&= \big(\sigma_j\sigma_i(a)\sigma_j(\mathrm{M}_i) + \sigma_j\delta_i(a)\big)\mathrm{M}_j + \big(\sigma_j\sigma_i(a)\delta_j(\mathrm{M}_i) + \delta_j\sigma_i(a)\mathrm{M}_i + \delta_j\delta_i(a)\big)\,{}^t\mathrm{B} \\
&= \big(\sigma_j\sigma_i(a)[\sigma_j(\mathrm{M}_i)\mathrm{M}_j + \delta_j(\mathrm{M}_i)] + [\sigma_j\delta_i(a)\mathrm{M}_j + \delta_j\sigma_i(a)\mathrm{M}_i + \delta_j\delta_i(a)]\big)\,{}^t\mathrm{B}.
\end{aligned}$$

Les commutations entre endomorphismes et σ-dérivations données par la définition des algèbres de Ore font que le dernier terme entre crochets est symétrique en i et j ; en tenant de plus compte, pour $i \neq j$, de la commutation $\partial_i\partial_j = \partial_j\partial_i$, on déduit la relation suivante, qui remplace la commutation entre les matrices M_ℓ du cas commutatif :

$$\sigma_j(\mathrm{M}_i)\mathrm{M}_j + \delta_j(\mathrm{M}_i) = \sigma_i(\mathrm{M}_j)\mathrm{M}_i + \delta_i(\mathrm{M}_j),$$

où la simplification par $\sigma_i\sigma_j(a)$ se justifie par l'hypothèse de surjectivité des σ_ℓ. Lorsque de telles relations sont assurées, la même méthode de recherche de dépendances linéaires par la méthode de Gauss que dans le cas commutatif s'applique et fournit un calcul de l'addition et du produit de fonctions ∂-finies, ou même directement d'une expression polynomiale en des fonctions ∂-finies, en faisant le bon choix de V. Dans la pratique, les matrices M_i sont le plus souvent obtenues à partir de bases de Gröbner : les b_ℓ correspondent à des points sous un escalier. Plutôt que de faire une présentation formelle de ces algorithmes, nous en donnons l'idée sur un exemple.

Exemple 31.5 Considérons le calcul du produit des deux fonctions f et g en deux variables x et y, données par $f(x,y) = \exp(xy)$ et $g(x,y) = \mathrm{J}_\mu(x+y)$, où, pour un paramètre μ complexe, J_μ est la fonction de Bessel de première espèce, solution de l'équation différentielle

$$z^2\mathrm{J}''_\mu(z) + z\mathrm{J}'_\mu(z) + (z^2-\mu^2)\mathrm{J}_\mu(z) = 0$$

qui admet à l'origine le développement asymptotique

$$\mathrm{J}_\mu(z) \sim \frac{1}{2^\mu}\sum_{n=0}^{\infty}\frac{(-1)^n(z/2)^{2n}}{n!\,\Gamma(n+\mu+1)}.$$

Considérons l'algèbre de Ore $\mathrm{A} = \mathbb{C}(\mu,x,y)\langle\partial_x,\partial_y;\mathrm{id},\mathrm{id},\mathrm{D}_x,\mathrm{D}_y\rangle$, où μ est maintenant un paramètre formel. Des bases de Gröbner des annulateurs I et J dans A de f et g pour l'ordre $\mathtt{lex}(\partial_y,\partial_x)$ sont respectivement

$$\{\partial_x - y, \partial_y - x\} \qquad \text{et} \qquad \{(x+y)^2\partial_x^2 + (x+y)\partial_x + (x+y)^2 - \mu^2, \partial_y - \partial_x\}.$$

On désigne encore maintenant par f et g les vecteurs cycliques générateurs des modules A/I et A/J, avec un petit abus de notation. Pour pouvoir parler d'un produit entre ces objets a priori seulement algébriques, on introduit l'espace vectoriel sur $\mathbb{C}(\mu)$ de base $\mathrm{B} = \big(f\otimes g, f\otimes(\partial_x\cdot g)\big)$, où l'on voit le produit $h = f\otimes g$ comme donné par ses coordonnées $(1,0)$. (On peut voir ce produit tensoriel comme un produit bilinéaire sans commutativité : en général, $u\otimes v \neq v\otimes u$, mais, pour tout $\lambda\in\mathbb{C}(\mu)$, $(\lambda u)\otimes v = u\otimes(\lambda v) = \lambda(u\otimes v)$, que l'on note donc par simplicité $\lambda u\otimes v$.) Sur cet espace de base B, qui n'est autre que $\mathrm{A/I}\otimes\mathrm{A/J}$, la dérivée d'un produit tensoriel est donnée par

$$\partial_x\cdot(u\otimes v) = (\partial_x\cdot u)\otimes v + u\otimes(\partial_x\cdot v),$$

ce que l'on peut résumer en disant que ∂_x agit par $\partial_x\otimes\mathrm{id} + \mathrm{id}\otimes\partial_x$, pour des domaines de définitions adéquats. Cette action fait du produit tensoriel $\mathrm{A/I}\otimes\mathrm{A/J}$ un nouveau module sur A. Puisque

$$\partial_x\cdot(f\otimes g) = (\partial_x\cdot f)\otimes g + f\otimes(\partial_x\cdot g) = yf\otimes g + f\otimes(\partial_x\cdot g)$$

et

$$\begin{aligned}\partial_x\cdot\big(f\otimes(\partial_x\cdot g)\big) &= yf\otimes(\partial_x\cdot g) + f\otimes(\partial_x^2\cdot g)\\ &= yf\otimes(\partial_x\cdot g) + \left(\frac{1}{(x+y)^2}\mu^2 - 1\right)f\otimes g - \frac{1}{x+y}f\otimes(\partial_x\cdot g),\end{aligned}$$

et des relations similaires pour l'action de ∂_y, on trouve les matrices

$$\mathrm{M}_x = \begin{pmatrix} y & 1 \\ \frac{\mu^2}{(x+y)^2} - 1 & y - \frac{1}{x+y} \end{pmatrix}$$

et

$$\mathrm{M}_y = \begin{pmatrix} x & 1 \\ \frac{\mu^2}{(x+y)^2} - 1 & x - \frac{1}{x+y} \end{pmatrix}.$$

Choisissons d'itérer selon un ordre raffinant le degré total en ∂_x et ∂_y, pour lequel $1 < \partial_y < \partial_x < \partial_y^2 < \partial_x\partial_y < \partial_x^2$, etc. On fait d'abord agir ∂_y pour trouver

$$\partial_y \cdot h = \left((1,0)\mathrm{M}_y + \frac{d}{dy}(1,0)\right) {}^t\mathrm{B} = (x,1)\,{}^t\mathrm{B},$$

qui avec $(1,0)\,{}^t\mathrm{B}$ ne constitue pas une famille liée. Ensuite, on trouve par le même procédé $\partial_x \cdot h = (y,1)\,{}^t\mathrm{B}$, ce qui fournit la liaison $p_1 \cdot h = 0$ pour $p_1 = \partial_x - \partial_y + (x-y)$. Pour la suite du calcul, on exclut alors tous les monômes divisibles par ∂_x ; le monôme considéré suivant est ∂_y^2. Son action sur h donne

$$\partial_y^2 \cdot h = \left((x,1)\mathrm{M}_y + \frac{d}{dy}(x,1)\right) {}^t\mathrm{B} = \left(\frac{\mu^2}{(x+y)^2} + x^2 - 1, 2x - \frac{1}{x+y}\right) {}^t\mathrm{B},$$

et l'on obtient un second annulateur de h,

$$p_2 = (x+y)^2\partial_y^2 - (x+y)(2x^2+2xy-1)\partial_y + (x+y)(x^3+x^2y+y) - \mu^2.$$

Pour la suite du calcul, on exclut donc tous les monômes divisibles par ∂_y^2, si bien qu'il ne reste plus aucun monôme à considérer. L'idéal annulateur de h est l'idéal $\mathrm{A}p_1 + \mathrm{A}p_2$, dont $\{p_1, p_2\}$ est une base de Gröbner pour l'ordre $\mathtt{tdeg}(\partial_y, \partial_x)$, de monômes de tête respectifs ∂_x et ∂_y^2 ; le module A/I est donné comme $\mathbb{C}(x,y)$-espace vectoriel par sa base $(1 + \mathrm{I}, \partial_x + \mathrm{I})$.

Le calcul qui précède peut se revisiter en abandonnant l'écriture matricielle et en faisant apparaître plus explicitement les calculs de restes modulo une base de Gröbner.

Exemple 31.6 On récrit d'abord $\partial_y \cdot h$ sous la forme

$$\partial_y \cdot h = (\partial_y \cdot f) \otimes g + f \otimes (\partial_y \cdot g) = xf \otimes g + f \otimes (\partial_x \cdot g),$$

après réductions par les bases de Gröbner pour I et J ; ce vecteur et h constituent une famille linéairement indépendante. On procède ensuite de même pour $\partial_x \cdot h$,

de façon à obtenir

$$\partial_x \cdot h = (\partial_x \cdot f) \otimes g + f \otimes (\partial_x \cdot g) = yf \otimes g + f \otimes (\partial_x \cdot g) \; ;$$

on retrouve ainsi l'annulateur p_1. Le monôme considéré suivant est ∂_y^2, d'action sur h

$$\partial_y^2 \cdot h = x^2 f \otimes g + 2xf \otimes (\partial_x \cdot g) - (x+y)^{-2} f \otimes \Big((x+y)\partial_x + (x+y)^2 - \mu^2\Big) g \; ;$$

ce vecteur, h et $\partial_y \cdot h$ forment une famille linéairement liée, et l'on retrouve le second annulateur p_2. Le calcul se termine de la même manière.

Pour l'algorithme d'addition, les mêmes idées algorithmiques fonctionnent en calculant dans la somme directe $A/I \oplus A/J$ au lieu de $A/I \otimes A/J$.

Notes

Le calcul de recherche de vecteur cyclique multivarié en Section 31.5 peut être vu comme une réminiscence de l'algorithme FGLM [Fau+93] pour le changement d'ordre dans les bases de Gröbner du cadre commutatif.

Les clôtures dans le cas à plusieurs variables ont été largement exploitées dans le traitement d'identités en fonctions spéciales, à la suite des travaux de Zeilberger [Zei90]. L'algorithmisation de ces méthodes s'est ensuite peu à peu fait jour, d'abord dans le cas différentiel [Tak92], puis dans le cas de polynômes tordus généraux [CS98], où la définition des fonctions ∂-finies a été posée. La présentation matricielle de la Section 31.5 nous a paru plus synthétique que la présentation classique à base de récriture par bases de Gröbner [CS98]. Les deux présentations trouvent leur inspiration dans l'algorithme FGLM [Fau+93].

Un autre cadre de bases de Gröbner est disponible pour des algèbres de Ore polynomiales, de la forme

$$k[x_1, \ldots, x_r]\langle \partial_1, \ldots, \partial_r; \sigma_1, \ldots, \sigma_r, \delta_1, \ldots, \delta_r \rangle$$

et non

$$k(x_1, \ldots, x_r)\langle \partial_1, \ldots, \partial_r; \sigma_1, \ldots, \sigma_r, \delta_1, \ldots, \delta_r \rangle.$$

Les ordres monomiaux garantissant la terminaison d'un algorithme de Buchberger sont alors sujet à plus de contraintes. Tout ceci est raconté de façon relativement accessible dans le livre de Saito, Sturmfels et Takayama [SST00], d'où est aussi tirée une partie de la Section 31.4. Le système différentiel de la Section 31.4 est un exemple d'une théorie développée par Gel'fand, Kapranov et Zelevinsky, connue sous le nom de système hypergéométrique GKZ.

Bibliographie

CS98 Chyzak, Frédéric et Bruno Salvy (1998). « Non-commutative elimination in Ore algebras proves multivariate holonomic identities ». In : *Journal of Symbolic Computation*, vol. 26, n°2, p. 187–227.

Fau+93 Faugère, J. C., P. Gianni, D. Lazard et T. Mora (1993). « Efficient computation of zero-dimensional Gröbner bases by change of ordering ». In : *Journal of Symbolic Computation*, vol. 16, n°4, p. 329–344.

SST00 Saito, Mutsumi, Bernd Sturmfels et Nobuki Takayama (2000). *Gröbner deformations of hypergeometric differential equations*. Springer-Verlag.

Tak92 Takayama, Nobuki (1992). « An approach to the zero recognition problem by Buchberger algorithm ». In : *Journal of Symbolic Computation*, vol. 14, n°2-3, p. 265–282.

Zei90 Zeilberger, Doron (1990). « A holonomic systems approach to special functions identities ». In : *Journal of Computational and Applied Mathematics*, vol. 32, n°3, p. 321–368.

32. Sommation et intégration symboliques des fonctions spéciales

Résumé

Nous décrivons un algorithme qui peut se voir comme une extension de l'algorithme de Zeilberger pour des sommants ∂-finis, et qui traite dans le même formalisme sommation et intégration. Les quelques sommes et intégrales suivantes, qui peuvent être traitées avec cet algorithme, montrent une variété d'applications qui vont de la combinatoire à la physique mathématique en passant par la théorie des fonctions spéciales :

$$\sum_{k=0}^{n}\left(\sum_{j=0}^{k}\binom{n}{j}\right)^3 = n2^{3n-1} + 2^{3n} - 3n2^{n-2}\binom{2n}{n},$$

$$\sum_{n=0}^{\infty} \mathrm{H}_n(x)\mathrm{H}_n(y)\frac{u^n}{n!} = \frac{\exp\left(\frac{4u(xy-u(x^2+y^2))}{1-4u^2}\right)}{\sqrt{1-4u^2}},$$

$$\frac{1}{2}\mathrm{J}_0(x)^2 + \mathrm{J}_1(x)^2 + \mathrm{J}_2(x)^2 + \cdots = \frac{1}{2},$$

$$\int_{-1}^{+1} \frac{e^{-px}\mathrm{T}_n(x)}{\sqrt{1-x^2}}\,dx = (-1)^n \pi \mathrm{I}_n(p),$$

$$\int_0^{+\infty} xe^{-px^2}\mathrm{J}_n(bx)\mathrm{I}_n(cx)\,dx = \frac{1}{2p}\exp\left(\frac{c^2-b^2}{4p}\right)\mathrm{J}_n\left(\frac{bc}{2p}\right),$$

$$\int_0^{+\infty} x\mathrm{J}_1(ax)\mathrm{I}_1(ax)\mathrm{Y}_0(x)\mathrm{K}_0(x)\,dx = -\frac{\ln(1-a^4)}{2\pi a^2},$$

$$\sum_{k=0}^{n}\frac{q^{k^2}}{(q;q)_k(q;q)_{n-k}} = \sum_{k=-n}^{n}\frac{(-1)^k q^{(5k^2-k)/2}}{(q;q)_{n-k}(q;q)_{n+k}}.$$

Ici, J, Y, I et K sont des variantes de fonctions de Bessel, lesquelles apparaissent fréquemment pour décrire des modèles physiques à symétrie cylindrique ou

sphérique ; H et T sont des familles de polynômes orthogonaux de Hermite et Tchebychev ; $(q;q)_n$ représente le produit $(1-q)\cdots(1-q^n)$. La première identité intervient dans une discrétisation d'une question de probabilités sur la position du maximum de trois variables aléatoires gaussiennes ; la dernière est une variante finie d'une des identités de Rogers–Ramanujan, en théorie des partitions.

32.1 Expression du télescopage créatif en termes d'algèbres de Ore rationnelles

La méthode du télescopage créatif a été exposée au Chapitre 29, Section 29.2, et appliquée à la sommation hypergéométrique définie, par une combinaison de l'algorithme de Gosper et d'une idée due à Zeilberger. Cette approche se généralise en des algorithmes de sommation et intégration pour les suites et fonctions ∂-finies.

Pour évaluer une somme paramétrée

$$\mathrm{F}_n = \sum_{k=a}^{b} f_{n,k},$$

le principe du télescopage créatif est de déterminer une suite auxiliaire $g = (g_{n,k})$ ainsi que des coefficients $\eta_0, \ldots, \eta_r$, fonctions de la variable n, tels que se trouve vérifiée la relation

$$\eta_r(n) f_{n+r,k} + \cdots + \eta_0(n) f_{n,k} = g_{n,k+1} - g_{n,k}.$$

Ici, nous ne faisons pas plus d'hypothèses sur les η_i et g que celle de pouvoir évaluer la relation ci-dessus pour tout n quand k décrit les entiers de a à b. Dans ce cas, une sommation sur k fournit l'égalité

$$\eta_r(n)\mathrm{F}_{n+r} + \cdots + \eta_0(n)\mathrm{F}_n = g_{n,b+1} - g_{n,a}.$$

Si le membre de droite n'est pas déjà nul, on recherche un opérateur annulateur de ce second membre ; par composition, on obtient une récurrence homogène sur F. Dans certains cas, on sait prédire à partir de conditions analytiques sur f la nullité du terme $g_{n,b+1} - g_{n,a}$.

Des algorithmes d'efficacités différentes ont été donnés selon le domaine de recherche des η_i et de g, et selon le compromis choisi entre efficacité et richesse de la classe de suites f en entrée. En particulier, l'Algorithme 29.2 de Zeilberger, page 537, optimisé pour une suite f hypergéométrique, revient à rechercher des η_i polynomiaux et une suite g similaire à f, c'est-à-dire un multiple ϕf pour une fraction rationnelle ϕ en n et k. La suite $g = \phi f$ devant être une somme indéfinie, la recherche de ϕ et des η_i se fait par une variante paramétrée de l'algorithme de Gosper (Section 29.1). Notons que le domaine de recherche de g est l'espace vectoriel $\mathbb{C}(n,k)f$, qui n'est autre, dans le cas hypergéométrique, que le module engendré par f sur l'algèbre de Ore $\mathrm{A} = \mathbb{C}(n,k)\langle \partial_n, \partial_k ; \mathrm{S}_n, \mathrm{S}_k \rangle$.

Nous considérons ici la généralisation au cas où f est une fonction ∂-finie et où le module $\mathrm{A}\cdot f$ est un espace vectoriel de dimension finie sur $\mathbb{C}(n,k)$, mais pas forcément de dimension 1. Soit $v_1, \ldots, v_d$ les éléments d'une base vectorielle de $\mathrm{A}\cdot f$;

l'algorithme de Zeilberger étendu qui va être expliqué en Section 32.2 et donné comme Algorithme 32.2 recherche g sous la forme indéterminée $\phi_1 v_1 + \cdots + \phi_d v_d$, pour des fractions rationnelles ϕ_i en n et k. Cette recherche se fait par une extension ∂-finie de la variante paramétrée de l'algorithme de Gosper, l'Algorithme 32.1.

Tout ce qui a été dit se transpose au monde différentiel pour l'évaluation d'une intégrale paramétrée

$$\mathrm{F}(x) = \int_a^b f(x,y)\,dy.$$

On cherche alors une relation

$$\eta_r(x)\frac{\partial^r f}{dx^r}(x,y) + \cdots + \eta_0(n) f(x,y) = \frac{\partial g}{dy}(x,y),$$

qui après intégration fournit l'égalité

$$\eta_r(n)\mathrm{F}^{(r)}(x) + \cdots + \eta_0(n)\mathrm{F}(x) = \int_a^b g(x,y)\,dy.$$

Un cas particulier important (par exemple pour la combinatoire) est celui dans lequel les « fonctions » f et F sont en réalité des séries formelles, car alors les questions de bonne définition des objets se simplifient. La même méthode permet aussi de traiter des sommations paramétrées continûment,

$$\mathrm{F}(x) = \sum_{k=a}^{b} f_k(x),$$

et des suites d'intégrales de la forme

$$\mathrm{F}_n = \int_a^b f_n(y)\,dy.$$

Exercice 32.1 Formuler la relation entre f et g à rechercher dans ces deux derniers cas. ∎

32.2 L'algorithme de sommation ∂-finie définie sur l'exemple $\frac{1}{2}\mathrm{J}_0(x)^2 + \mathrm{J}_1(x)^2 + \mathrm{J}_2(x)^2 + \cdots = \frac{1}{2}$

Nous allons montrer que la famille paramétrée des fonctions de Bessel de première espèce, J_ν, où chaque J_ν est une solution que nous allons préciser de l'équation de Bessel

$$x^2 y''(x) + xy'(x) + (x^2 - \nu^2)y(x) = 0, \tag{32.1}$$

a une somme $\frac{1}{2}\mathrm{J}_0(x)^2 + \mathrm{J}_1(x)^2 + \mathrm{J}_2(x)^2 + \cdots$ qui s'évalue à $\frac{1}{2}$.

L'équation de Bessel et les fonctions de Bessel peuvent être considérées pour des valeurs complexes du paramètre ν, mais vu la nature de la somme à étudier, nous nous limiterons dorénavant à des valeurs entières $\nu \in \mathbb{N}$. La traduction de l'équation

de Bessel selon la méthode de la Section 17.1, puis l'étude de son polynôme indiciel à l'origine (Définition 17.1), montrent qu'il existe pour chaque ν des solutions dans les séries formelles $\mathbb{C}[[x]]$ et que ces solutions constituent un espace vectoriel de dimension 1 sur $\mathbb{C}$ de séries. Une base de ces solutions formelles est donnée par la série de Bessel

$$\mathrm{J}_\nu(x) = \left(\frac{x}{2}\right)^\nu \sum_{n=0}^{\infty} \frac{(-1)^n}{n!\,(n+\nu)!} \left(\frac{x}{2}\right)^{2n},$$

de valuation ν. Vu la décroissance des coefficients, cette série s'interprète aussi pour chaque entier ν comme une série de fonctions convergente.

Exercice 32.2 Vérifier ces résultats. ∎

On vérifie par simple substitution et évaluation que ces fonctions J_ν satisfont en plus de (32.1) aux relations

$$x\mathrm{J}'_\nu(x) + x\mathrm{J}_{\nu+1}(x) - \nu\mathrm{J}_\nu(x) = 0 \qquad \text{et} \qquad x\mathrm{J}_{\nu+2}(x) - 2(\nu+1)\mathrm{J}_{\nu+1}(x) + x\mathrm{J}_\nu(x) = 0.$$

En introduisant l'algèbre de Ore $\mathrm{A} = \mathbb{C}(\nu, x)\langle \partial_\nu, \partial_x; \mathrm{S}_\nu, \mathrm{id}, 0, \mathrm{D}_x\rangle$ où S_ν est le décalage avant sur ν et D_x est la dérivation par rapport à x, on a donc un système d'annulateurs pour J,

$$\begin{aligned} p_1 &= x^2\partial_x^2 + x\partial_x + x^2 - \nu^2, \\ p_2 &= x\partial_x + x\partial_\nu - \nu, \\ p_3 &= x\partial_\nu^2 - 2(\nu+1)\partial_\nu + x. \end{aligned}$$

Les deux premiers forment une base de Gröbner de l'idéal engendré par (p_1, p_2, p_3) pour l'ordre $\mathrm{lex}(\partial_\nu, \partial_x)$; les deux derniers pour l'ordre $\mathrm{lex}(\partial_x, \partial_\nu)$.

Exercice 32.3 Pour chacun des idéaux $\mathrm{A}p_1 + \mathrm{A}p_2$ et $\mathrm{A}p_2 + \mathrm{A}p_3$, calculer la base de Gröbner minimale réduite pour chacun des deux ordres $\mathrm{lex}(\partial_\nu, \partial_x)$ et $\mathrm{lex}(\partial_x, \partial_\nu)$. ∎

Ainsi, J est une fonction ∂-finie. Le module $\mathrm{A}\cdot\mathrm{J}$ est donné, par exemple, comme l'espace vectoriel sur $\mathbb{C}(\nu, x)$ de base $(\mathrm{J}, \partial_\nu \cdot \mathrm{J})$. Pour représenter le carré de J en vue d'une sommation, on peut observer que, en tant qu'espace vectoriel, le module $\mathrm{A}\cdot\mathrm{J}^2$ admet la base $\left(\mathrm{J}^2, \mathrm{J}\times(\partial_\nu\cdot\mathrm{J}), (\partial_\nu\cdot\mathrm{J})^2\right)$ et utiliser l'algorithme de clôture par produit pour obtenir une base de Gröbner. En fait, le calcul qui suit n'a même pas besoin d'une représentation aussi explicite de $f = \mathrm{J}^2$: pour calculer la somme $\frac{1}{2}\mathrm{J}_0(x)^2 + \mathrm{J}_1(x)^2 + \mathrm{J}_2(x)^2 + \cdots$ comme fonction de x, on recherche (quand on en vient à un ordre cible r égal à 1) une fonction η de x, indépendante de ν, telle que $f' + \eta f$ soit la différence finie en ν d'un élément g de $\mathrm{A}\cdot\mathrm{J}^2$. Pour la suite du calcul, nous fixons cet élément sous la forme indéterminée donnée par

$$g(\nu) = \phi_0(\nu)\mathrm{J}_\nu^2 + \phi_1(\nu)\mathrm{J}_{\nu+1}^2 + \phi_2(\nu)\mathrm{J}_\nu\mathrm{J}_{\nu+1},$$

où nous avons omis de faire référence à la variable x dans les évaluations de g, des ϕ_i et de J, car cette variable ne va intervenir que comme paramètre dans le calcul des fractions rationnelles ϕ_i. (On peut penser qu'on travaille temporairement dans l'algèbre de Ore $\mathrm{A}' = \mathbb{C}(\nu, x)\langle \partial_\nu; \mathrm{S}_\nu\rangle$.)

En supposant le problème résolu, on a alors par construction la relation $f' + \eta f = (\partial_\nu - 1)\cdot g$, puis, après réduction de chaque occurence des dérivées et décalées de J par la base de Gröbner $\{p_2, p_3\}$,

$$\begin{aligned}2\mathrm{J}_\nu\mathrm{J}'_\nu + \eta\mathrm{J}_\nu^2 &= (\partial_\nu - 1)\cdot\Big(\phi_0(\nu)\mathrm{J}_\nu^2 + \phi_1(\nu)\mathrm{J}_{\nu+1}^2 + \phi_2(\nu)\mathrm{J}_\nu\mathrm{J}_{\nu+1}\Big)\\ &= \phi_0(\nu+1)\mathrm{J}_{\nu+1}^2 - \phi_0(\nu)\mathrm{J}_\nu^2 + \phi_1(\nu+1)x^{-2}\Big(2(\nu+1)\mathrm{J}_{\nu+1} - x\mathrm{J}_\nu\Big)^2 - \phi_1(\nu)\mathrm{J}_{\nu+1}^2\\ &\qquad + \phi_2(\nu+1)x^{-1}\mathrm{J}_{\nu+1}\Big(2(\nu+1)\mathrm{J}_{\nu+1} - x\mathrm{J}_\nu\Big) - \phi_2(\nu)\mathrm{J}_\nu\mathrm{J}_{\nu+1},\end{aligned}$$

laquelle se récrit

$$\begin{aligned}&\Big(2\nu x^{-1} + \eta\Big)\mathrm{J}_\nu^2 - 2\mathrm{J}_\nu\mathrm{J}_{\nu+1}\\ &\quad= \Big(\phi_1(\nu+1) - \phi_0(\nu)\Big)\mathrm{J}_\nu^2 - \Big(4(\nu+1)x^{-1}\phi_1(\nu+1) + \phi_2(\nu+1) + \phi_2(\nu)\Big)\mathrm{J}_\nu\mathrm{J}_{\nu+1}\\ &\qquad + \Big(\phi_0(\nu+1) + 4(\nu+1)^2x^{-2}\phi_1(\nu+1) - \phi_1(\nu) + 2(\nu+1)x^{-1}\phi_2(\nu+1)\Big)\mathrm{J}_{\nu+1}^2.\end{aligned}$$

De l'indépendance linéaire des fonctions J_ν^2, $\mathrm{J}_{\nu+1}^2$ et $\mathrm{J}_\nu\mathrm{J}_{\nu+1}$ sur $\mathbb{C}(\nu, x)$, on déduit les relations nécessaires

$$\begin{aligned}-\phi_0(\nu) + \phi_1(\nu+1) &= 2\nu x^{-1} + \eta,\\ 4(\nu+1)x^{-1}\phi_1(\nu+1) + \phi_2(\nu) + \phi_2(\nu+1) &= 2,\\ \phi_0(\nu+1) - \phi_1(\nu) + 4(\nu+1)^2x^{-2}\phi_1(\nu+1) + 2(\nu+1)x^{-1}\phi_2(\nu+1) &= 0.\end{aligned}$$

En résolvant les deux premières respectivement en ϕ_0 et en ϕ_1, puis en substituant dans la dernière, on trouve la récurrence

$$\begin{aligned}&x^2(\nu+1)\phi_2(\nu+3) - (\nu+1)(4\nu^2 + 20\nu + 24 - x^2)\phi_2(\nu+2)\\ &\quad + (\nu+3)(4\nu^2 + 12\nu + 8 - x^2)\phi_2(\nu+1) - x^2(\nu+3)\phi_2(\nu) = -4x(\eta\nu^2 + 4\eta\nu + 3\eta + x).\end{aligned}$$

Nous résolvons maintenant celle-ci en ses solutions rationnelles par l'algorithme d'Abramov (Section 16.4), dans sa variante paramétrée qui résoud en $\phi_2 \in \mathbb{C}(n, \nu)$ et simultanément en $\eta \in \mathbb{C}$, par l'algorithme de la Section 16.4 du même chapitre. Les coefficients extrêmes de la partie homogène indiquent que toute solution rationnelle doit être polynomiale, puisque le pgcd entre $(\nu-3)+1$ et $\nu+3+h$ vaut 1 pour tout $h \in \mathbb{N}$ (voir l'Algorithme 16.6, en page 304, pour le calcul d'un multiple du dénominateur). En faisant apparaître une différence finie dans la partie homogène, on montre que l'opérateur associé accroît de 2 le degré d'un polynôme. Le membre gauche de la récurrence se récrit en effet :

$$\begin{aligned}&x^2(\nu+1)\phi_2(\nu+3) - (12\nu^2 + 60\nu + 72 - 3x^2)\phi_2(\nu+2)\\ &\qquad - (\partial_\nu - 1)\cdot\Big((\nu+3)(4\nu^2 + 12\nu + 8 - x^2)\phi_2(\nu+1)\Big) - x^2(\nu+3)\phi_2(\nu),\end{aligned}$$

où l'on voit que pour un polynôme $\phi_2(n)$, l'unique terme de plus haut degré est celui porté par $\phi_2(n+2)$. Le degré de la partie inhomogène étant 2, toute solution

Entrée Un terme ∂-fini $f(x)$ par rapport à $A = \mathbb{K}(x)\langle\partial;\sigma,\delta\rangle$, un opérateur annulateur L de f, d'ordre s.
Sortie Un terme ∂-fini $g(x)$ tel que, respectivement, $g'(x) = f(x)$ dans le cas différentiel, ou $g(x+1) - g(x) = f(x)$ dans le cas à récurrence, ou encore le symbole $\nexists$.

1. Poser $\Phi = \phi_{s-1}\partial_x^{s-1} + \cdots + \phi_0$ pour des coefficients indéterminés $\phi_i \in \mathbb{K}(x)$.
2. Calculer le reste Z de la division euclidienne par L de, respectivement, $\partial_x\Phi - 1$ dans le cas différentiel, ou $(\partial_x - 1)\Phi - 1$ dans le cas à récurrence.
3. Extraire le système sur les ϕ_i obtenu en annulant les s coefficients de Z.
4. Découpler le système et le résoudre en les ϕ_i par les algorithmes du Chapitre 16.
5. Si la résolution fournit une solution, renvoyer $\Phi \cdot f$, sinon renvoyer $\nexists$.

Algorithme 32.1 – Algorithme de sommation/intégration ∂-finie indéfinie.

rationnelle ne peut être qu'une constante. On trouve $\phi_2 = 1$ et $\eta = 0$, d'où après report $\phi_1 = 0$ et $\phi_0 = -2\nu/x$. Autrement dit, on a

$$\partial_x \cdot \mathrm{J}_\nu^2 = (\partial_\nu - 1) \cdot \left(\mathrm{J}_\nu \mathrm{J}_{\nu+1} - 2\nu x^{-1}\mathrm{J}_\nu^2\right),$$

qui par sommation fournit

$$\partial_x \cdot \sum_{\nu=0}^{\mathrm{N}} \mathrm{J}_\nu^2 = \mathrm{J}_{\mathrm{N}+1}\left(\mathrm{J}_{\mathrm{N}+2} - 2(\mathrm{N}+1)x^{-1}\mathrm{J}_{\mathrm{N}+1}\right) - \mathrm{J}_0\mathrm{J}_1.$$

Comme la série $\mathrm{J}_\mathrm{N} \in \mathbb{C}[[x]]$ a valuation N, le membre droit tend vers $-\mathrm{J}_0\mathrm{J}_1 = \frac{1}{2}\partial_x \cdot \mathrm{J}_0^2$ quand N tend vers ∞ pour la topologie usuelle donnée par la métrique $|s| = 2^{-v}$ pour toute série non nulle s de valuation v. On a donc

$$\partial_x \cdot \left(\tfrac{1}{2}\mathrm{J}_0(x)^2 + \mathrm{J}_1(x)^2 + \mathrm{J}_2(x)^2 + \cdots\right) = 0$$

qui caractérise la somme par une condition initiale. Une simple évaluation en 0 montre que la somme vaut $\frac{1}{2}$, ce qui achève la preuve de l'identité annoncée.

32.3 Description des algorithmes de sommation et intégration ∂-finies

L'Algorithme 29.1 de Gosper, page 532, pour la sommation hypergéométrique indéfinie, se généralise assez directement en un algorithme pour la sommation et l'intégration ∂-fini indéfinie, la différence essentielle étant que les objets du calcul ne sont plus paramétrés par une unique fraction rationnelle, mais maintenant par une collection de fractions rationnelles indicées par un escalier de base de Gröbner. L'Algorithme 32.1 en résulte.

Entrée Un terme ∂-fini $u(x,y)$ par rapport à $A = \mathbb{K}(x,y)\langle \partial_x, \partial_y; \sigma_x, \sigma_y, \delta_x, \delta_y\rangle$, une base de Gröbner G de l'idéal annulateur de u (pour un certain ordre monomial), un entier $m \in \mathbb{N}$.

Sortie Une famille $(\eta_i(x))_{0\le i\le r}$ de fractions rationnelles de $\mathbb{K}(x)$, ainsi qu'une famille $\phi_{(\alpha,\beta)}(x,y)$ de fractions rationnelles de $\mathbb{K}(x,y)$, indexée par les monômes sous l'escalier de G, telles que

$$\eta_r(x)\partial_x^r u(x,y) + \cdots + \eta_0(x)u(x,y) = \Delta_y \sum_{\alpha,\beta} \phi_{\alpha,\beta}(x,y)\partial_x^\alpha \partial_y^\beta u(x,y),$$

(où Δ_y vaut ∂_y dans le cas de l'intégration par rapport à y et $\partial_y - 1$ dans le cas de la sommation sur y), ou alors, le symbole $\nexists$ pour indiquer qu'il n'existe pas de telle famille.

1. Pour r de 0 à m :
   a. calculer le reste H_i de la division de ∂_x^i par G ;
   b. poser $\Phi = \sum_{\alpha,\beta} \phi_{\alpha,\beta}(y)\partial_x^\alpha \partial_y^\beta$ où (α,β) décrit les monômes sous l'escalier de G, pour des coefficients indéterminés $\phi_{\alpha,\beta} \in \mathbb{K}(x,y)$;
   c. calculer le reste Z de la division par G de
   $$\Delta_y\Phi - \sum_{i=0}^{r} \eta_i(x)H_i(x,y,\partial_x,\partial_y);$$
   d. extraire le système obtenu en annulant les coefficients de $\partial_x^\alpha\partial_y^\beta$ quand (α,β) décrit les monômes sous l'escalier de G ;
   e. découpler le système et le résoudre en les $\phi_{\alpha,\beta}$ et les η_i par les versions paramétrées des algorithmes du Chapitre 16 ;
   f. si la résolution fournit une solution, renvoyer la famille $(\eta_i(x))$ et la famille $(\phi_{\alpha,\beta}(x,y))$.
2. Renvoyer $\nexists$.

Algorithme 32.2 – Algorithme de sommation/intégration ∂-finie définie.

À la différence du cas hypergéométrique, l'algorithme de Gosper étendu ne va pas être utilisé aussi directement dans l'algorithme de Chyzak pour la sommation et l'intégration ∂-finies définies, lequel est donné pour le cadre général par l'Algorithme 32.2. Mais l'algorithme de Gosper étendu (Algorithme 32.1) peut être vu comme une version simplifiée des étapes (1.b)–(1.f) de l'Algorithme 32.2 de Chyzak lorsque le dessous d'escalier est en fait donné par des monômes successifs en une

seule dérivation. Le découplage à l'étape (1.e) peut être effectué par calcul de base de Gröbner de modules, comme indiqué ci-dessous à la Section 32.4. Comme dans le cas de l'algorithme de Zeilberger, il est intéressant de discuter la question de la terminaison d'une variante de l'Algorithme 32.2 à laquelle on ne fournirait pas de borne m sur r : dans le cas différentiel, une solution (ϕ, η) existe toujours pour un certain r, l'algorithme avec une boucle ouverte termine toujours dans ce cas ; pour les récurrences, une hypothèse supplémentaire est nécessaire pour garantir l'existence d'une solution, et l'algorithme présenté peut donc échouer en renvoyant $\nexists$, soit parce qu'il n'y a pas de solution, soit parce que la borne m a été choisie trop petite.

32.4 Bases de Gröbner de modules et découplage de systèmes

Dans l'exemple qui précède, on a pour le moment effectué le découplage « à la main », mais un procédé systématique et automatique est disponible, par le biais des bases de Gröbner de modules, lesquelles généralisent la notion de base de Gröbner pour les idéaux. Cette notion existe tant dans le domaine des polynômes commutatifs que dans le cadre non commutatif des algèbres de Ore ; nous la présentons directement dans ce second cas.

Dans le cas d'un idéal I d'une algèbre de Ore A, les éléments de I s'interprètent comme autant d'équations vérifiées par une fonction inconnue ϕ. Dans une perspective algorithmique, chaque idéal est donné par un nombre fini de générateurs. Une question naturelle est celle de systèmes linéaires sur un vecteur de fonctions inconnues $(\phi_1, \ldots, \phi_d)$, à coefficients dans A. On considère des systèmes d'un nombre fini d'équations de la forme $g_i = g_{i,1} \cdot \phi_1 + \cdots + g_{i,d} \cdot \phi_d$ pour des polynômes tordus $g_{i,j}$ de A. Un tel système se représente de façon compacte par une matrice $(g_{i,j})$ à entrées dans A. Les questions qui sont alors naturelles sont celles de l'algèbre linéaire pour ces matrices, dont en particulier celle de donner un algorithme du pivot de Gauss pour des coefficients dans A, c'est-à-dire non plus dans un corps, mais dans un anneau, et non commutatif de surcroît.

Pour ce faire, au lieu de considérer simplement un ordre monomial sur les monômes ∂^a d'une algèbre de Ore $\mathrm{A} = k(x)\langle\partial; \sigma, \delta\rangle$, pour lequel tout idéal à gauche admet une base de Gröbner (voir le Théorème 31.1), on s'intéresse plus généralement à un module libre de rang fini sur A (voir la définition en Section 30.1), donné par une base sous la forme $\mathrm{A}^d = \mathrm{A}e_1 + \cdots + \mathrm{A}e_d$ et muni d'un ordre sur les $\partial^a e_i$, dans lequel on va étendre la notion de base de Gröbner pour des sous-modules à gauche de A^d. La notion d'ordre monomial conserve formellement la même définition, si ce n'est que les e_i ne peuvent apparaître que linéairement dans les monômes $\partial^a e_i$ et qu'ils ne peuvent servir pour des multiplications à gauche. La notion de S-polynôme s'étend aussi mot pour mot, à ceci près que deux polynômes de monômes de tête $\partial^a e_i$ et $\partial^b e_j$ ont un S-polynôme nul dès lors que i et j sont différents. Les définitions et caractérisations équivalentes des bases de Gröbner d'idéaux restent alors valables pour les sous-modules du module libre A^d. L'algorithme de Buchberger, modifié pour suivre ces nouvelles définitions, termine sur tout sous-module en fournissant une base de Gröbner. Pour certains ordres, ce calcul correspond à l'algorithme de Gauss.

Exercice 32.4 Donner des exemples d'ordres qui font que l'algorithme de Buchberger pour les modules simule une réduction par l'algorithme de Gauss. ∎

Un point de vue presque équivalent, mais qui donne une variante des calculs avec un peu plus de réductions à zéro (calculs inutiles, donc), est que le calcul est celui d'une base de Gröbner dans l'anneau $A[e_1,\dots,e_d]$ des polynômes en les indéterminées commutatives e_i à coefficients dans l'anneau A pour l'idéal à gauche engendré par les g_i initiaux et tous les produits $e_ie_j = 0$.

Reprenons l'exemple du découplage des relations entre les coordonnées ϕ_i donnant g dans la section précédente sur la somme des carrés des fonctions de Bessel. Ces relations se recodent par les éléments

$$\begin{aligned}
g_1 &:= -e_0 + \partial_\nu e_1 - (2\nu x^{-1} + \eta)e_3,\\
g_2 &:= 4(\nu+1)x^{-1}\partial_\nu e_1 + (\partial_\nu + 1)e_2 - 2e_3,\\
g_3 &:= \partial_\nu e_0 + \Big(4(\nu+1)^2x^{-2}\partial_\nu - 1\Big)e_1 + 2(\nu+1)x^{-1}\partial_\nu e_2,\\
g_4 &:= (\partial_\nu - 1)e_3,
\end{aligned}$$

du module libre A^4 pour l'algèbre de Ore $A = \mathbb{C}(n,k)\langle\partial_n,\partial_k;S_n,S_k\rangle$. Ici, chaque e_i représente la fonction rationnelle inconnue ϕ_i, et on a représenté les second membres des équations inhomogènes d'origine comme multiples d'une nouvelle inconnue représentée par e_3 et contrainte par g_4 à être constante.

Le découplage effectué dans la section précédente revient au calcul d'une base de Gröbner pour l'ordre $1ex(e_0,e_1,e_2,e_3,\partial_\nu)$. Les monômes de tête respectifs des g_i sont e_0, $\partial_\nu e_1$, $\partial_\nu e_0$ et $\partial_\nu e_3$, si bien que le seul S-polynôme non nul est $Spoly(g_1,g_3)$. Il est donné par

$$\begin{aligned}
Spoly(g_1,g_3) &= \partial_\nu g_1 + g_3\\
&= \Big(\partial_\nu^2 + 4(\nu+1)^2x^{-2}\partial_\nu - 1\Big)e_1 + 2(\nu+1)x^{-1}\partial_\nu e_2 - \Big(2(\nu+1)x^{-1} + \eta\Big)\partial_\nu e_3.
\end{aligned}$$

Après réductions par g_2 et g_3, ce polynôme devient

$$g_5 = -e_1 - \left(\frac{x}{4(\nu+2)}\partial_\nu^2 + \frac{x^2 - 4\nu^2 - 12\nu - 8}{4x(\nu+2)}\partial_\nu + \frac{\nu+1}{x}\right)e_2 + \left(\frac{x}{2(\nu+2)} - \eta\right)e_3,$$

qui est adjoint à la base de Gröbner en cours de calcul. L'unique nouvel S-polynôme à considérer est celui entre ce g_5 et g_2, qui est $Spoly(g_2,g_4) = g_2 + 4(\nu+1)x^{-1}\partial_\nu g_5$ et a pour monôme de tête $\partial_\nu^3 e_2$. Après réduction par e_3 et renormalisation, le dernier polynôme introduit dans la base de Gröbner est

$$\begin{aligned}
\Big((\nu+1)x^2\partial_\nu^3 + (\nu+1)(x^2 - 4\nu^2 - 20\nu - 24)\partial_\nu^2\\
- (\nu+3)(x^2 - 4\nu^2 - 12\nu - 8)\partial_\nu - (\nu+3)x^2\Big)e_2\\
+ 4\Big((\nu^2 + 4\nu + 3)\eta + x\Big)xe_3.
\end{aligned}$$

Ce polynôme n'est autre qu'un recodage de l'équation inhomogène du troisième ordre qui a permis de déterminer ϕ_2 dans la section précédente.

Notes

Dans les implantations de l'Algorithme 32.2 de Chyzak [Chy00], le découplage effectué à l'étape (1.e) est une étape coûteuse et la cause de lenteurs dans la résolution qui suit. Une approche alternative qui n'a jamais été étudiée en profondeur serait l'utilisation d'algorithmes pour la résolution directe de systèmes en leurs solutions rationnelles [Bar99]. Malgré tout, il n'est pas clair qu'une telle approche directe doive être meilleure en complexité ; un premier pas dans la direction d'une étude en complexité a été fait par Bostan, Chyzak et Panafieu avec l'analyse de la complexité du découplage [BCP13]. Une variante heuristique de l'algorithme de Chyzak a été donnée par Koutschan [Kou10]. Bien qu'elle ne garantisse pas de trouver un résultat (y compris dans le cas différentiel !), cette variante a plusieurs avantages : elle évite tout découplage et prédit très souvent en pratique les bons dénominateurs pour les solutions $\phi_{\alpha,\beta}$, tout en étant extrêmement efficace.

Bibliographie

Bar99 Barkatou, Moulay A. (1999). « On rational solutions of systems of linear differential equations ». In : *Journal of Symbolic Computation*, vol. 28, n°4-5. Differential algebra and differential equations, p. 547–567.

BCP13 Bostan, Alin, Frédéric Chyzak et Élie de Panafieu (2013). « Complexity estimates for two uncoupling algorithms ». In : *ISSAC 2013, Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation*. ACM, New York, p. 85–92.

Chy00 Chyzak, Frédéric (2000). « An extension of Zeilberger's fast algorithm to general holonomic functions ». In : *Discrete Mathematics*, vol. 217, n°1-3, p. 115–134.

Kou10 Koutschan, Christoph (2010). « A fast approach to creative telescoping ». In : *Mathematics in Computer Science*, vol. 4, n°2-3, p. 259–266.

Annexes

33. Exercices récapitulatifs

Sauf mention contraire, dans toute la suite, $\mathbb{K}$ désigne un corps effectif de caractéristique nulle.

Exercice 33.1 — Équivalence de complexités de diverses opérations polynomiales. Soit $\mathbb{K}$ un corps de caractéristique différente de 2 et soient $\mathsf{M}(n), \mathsf{I}(n), \mathsf{D}(n)$ et $\mathsf{S}(n)$ les complexités (mesurées en nombre d'opérations arithmétiques dans $\mathbb{K}$) pour calculer respectivement : le produit de deux polynômes de $\mathbb{K}[\mathrm{X}]$ de degré $< n$, l'inverse modulo X^n d'un polynôme de degré $< n$, le reste de la division d'un polynôme de degré $< 2n$ par un polynôme de degré n et le carré d'un polynôme de degré $< n$. Le but de l'exercice est de prouver que les fonctions $\mathsf{M}, \mathsf{I}, \mathsf{D}$ et S ont le même ordre de grandeur.

1. Par quelle technique peut-on montrer que $\mathsf{I} \in \mathrm{O}(\mathsf{M})$ et $\mathsf{D} \in \mathrm{O}(\mathsf{M})$?
2. Prouver l'identité $y^2 = \left(y^{-1} - (y+1)^{-1}\right)^{-1} - y$ et conclure que $\mathsf{S} \in \mathrm{O}(\mathsf{I})$.
3. Montrer que $\mathsf{M} \in \mathrm{O}(\mathsf{S})$, en utilisant l'égalité $2fg = (f+g)^2 - f^2 - g^2$.
4. Le polynôme réciproque d'un polynôme $b \in \mathbb{K}[\mathrm{X}]$ de degré n est noté $\mathrm{réc}_n(b)$. Établir une relation entre $\mathrm{réc}_n(b)^{-1} \bmod \mathrm{X}^n$ et le quotient de la division de X^{2n} par b. En déduire que $\mathsf{I} \in \mathrm{O}(\mathsf{D})$.
5. Conclure que $\mathrm{O}(\mathsf{M}) = \mathrm{O}(\mathsf{I}) = \mathrm{O}(\mathsf{S}) = \mathrm{O}(\mathsf{S})$.

■

Exercice 33.2 — Décalage rapide de polynômes.

1. Montrer que si P est un polynôme dans $\mathbb{Q}[\mathrm{X}]$ de degré au plus n, alors, quel que soit $a \in \mathbb{Q}$, on peut calculer les coefficients du « polynôme décalé » $\mathrm{Q}(\mathrm{X}) = \mathrm{P}(\mathrm{X}+a)$ en $\mathrm{O}(\mathsf{M}(n))$ opérations dans $\mathbb{Q}$.

Indication : commencer par prouver l'identité suivante dans $\mathbb{Q}[X]$:

$$\sum_{i=0}^{n} P^{(i)}(a)X^{n-i} = \left(\sum_{i=0}^{n} P^{(i)}(0)X^{n-i}\right) \times \left(\sum_{i=0}^{n} \frac{(aX)^i}{i!}\right) \bmod X^{n+1}.$$

2. Montrer que les coefficients des numérateurs et des dénominateurs des fractions rationnelles

$$R_0 = \frac{1}{X-1} + \frac{1}{X-2} + \cdots + \frac{1}{X-n} \quad \text{et} \quad R_1 = \frac{1}{X-1} + \frac{2}{X-2} + \cdots + \frac{n}{X-n}$$

peuvent être calculés en $O(M(n))$ opérations dans $\mathbb{Q}$.

Indication : pour R_0, donner un algorithme de type « diviser pour régner » basé sur l'algorithme de la question (1) ; pour R_1, en ramener le calcul à celui de R_0.

La formule de Lagrange montre qu'il existe un unique polynôme $P(X) \in \mathbb{Q}[X]$ de degré strictement inférieur à n tel que

$$P(i) = (-1)^i \binom{n}{i-1}^{-1}, \quad \text{pour tout} \quad i = 1, 2, \ldots, n.$$

3. Exprimer P en termes de R_0 et R_1 et donner un algorithme qui calcule les coefficients de P en $O(M(n))$ opérations dans $\mathbb{Q}$. Ces coefficients sont supposés connus dans la suite de l'exercice.
4. Montrer que la suite d'éléments de $\mathbb{Q}$ de terme général $a_i = P(i)$ vérifie une récurrence linéaire d'ordre n à coefficients constants.
5. En déduire un algorithme pour calculer les valeurs $P(n+1), \ldots, P(2n)$ en $O(M(n))$ opérations dans $\mathbb{Q}$.
6. Montrer qu'on peut calculer $P(n+1), \ldots, P(2n)$ en $O(n)$ opérations.

■

Exercice 33.3 — Extrapolation sur une suite arithmétique. Soit P un polynôme inconnu de $\mathbb{K}[X]$. Le but de l'exercice est de montrer que, à partir de la donnée d'un élément $a \in \mathbb{K}$, d'une borne d sur le degré de P, et des valeurs $P(0), P(1), \ldots, P(d)$, il est possible de déterminer l'ensemble des valeurs $P(a), P(a+1), \ldots, P(a+d)$ en $O(M(d))$ opérations dans $\mathbb{K}$.

1. Traiter d'abord le cas où a appartient à l'ensemble $\{-d, -d+1, \ldots, d-1, d\}$.

Supposons dans la suite que $a \in \mathbb{K}$ est tel que tous les éléments $a-d, \ldots, a+d$ sont non nuls. On rappelle la formule d'interpolation de Lagrange

$$P(X) = \sum_{i=0}^{d} p_i \prod_{j=0,\, j\neq i}^{d} (X-j), \quad \text{où} \quad p_i = \frac{P(i)}{\prod_{j=0,\, j\neq i}^{d} (i-j)}. \tag{L}$$

2. Montrer que les éléments $p_0, p_1, \ldots, p_d$ peuvent être tous calculés en $O(d)$ opérations dans $\mathbb{K}$.

En substituant $X = a + k$ dans l'équation (L), on obtient l'égalité

$$P(a+k) = \Delta_k \cdot \sum_{i=0}^{d} \frac{p_i}{a+k-i}, \quad \text{où} \quad \Delta_k = \prod_{j=0}^{d} (a+k-j). \tag{$\tilde{L}$}$$

3. Montrer que les éléments Δ_k pour $k = 0, \ldots, d$, peuvent être tous calculés en $O(d)$ opérations dans $\mathbb{K}$.
4. Avec l'égalité ($\tilde{L}$), conclure que l'ensemble des valeurs $P(a), P(a+1), \ldots, P(a+d)$ peut se calculer en $O(M(d))$ opérations dans $\mathbb{K}$.

■

Exercice 33.4 — Décomposition en éléments simples. On considère une fraction rationnelle

$$F(X) = \frac{P(X)}{Q_1^{\ell_1}(X) \cdots Q_k^{\ell_k}(X)},$$

où $P, Q_1, \ldots, Q_k$ sont des polynômes de $\mathbb{Q}[X]$ et les ℓ_i sont des entiers strictement positifs tels que
(H1) $\deg(P) < \sum_{i=1}^{k} \ell_i \cdot \deg(Q_i) = n$, et
(H2) $Q_1, \ldots, Q_k$ sont deux à deux premiers entre eux.
La décomposition en éléments simples de F est une écriture

$$\frac{P(X)}{Q_1^{\ell_1}(X) \cdots Q_k^{\ell_k}(X)} = \sum_{i=1}^{k} \sum_{j=1}^{\ell_i} \frac{C_{i,j}(X)}{Q_i^j(X)},$$

où les $C_{i,j}$ sont des polynômes de $\mathbb{Q}[X]$ tels que $\deg(C_{i,j}) < \deg(Q_i)$ pour tous i, j. L'existence et l'unicité des polynômes $C_{i,j}$ est admise. On appelle DES_n le problème du calcul des polynômes $C_{i,j}$ à partir des polynômes $P, Q_1, \ldots, Q_k$ et des entiers $n, \ell_1, \ldots, \ell_k$ satisfaisant aux hypothèses (H1) et (H2).

L'objectif de cet exercice est de montrer qu'il est possible de résoudre le problème DES_n en $O(M(n) \log n)$ opérations. Ici, et dans toute la suite de l'exercice, par *opération* on entend *opération arithmétique dans le corps* $\mathbb{Q}$ et $M(n)$ représente une borne sur le nombre d'opérations suffisant à multiplier deux polynômes de $\mathbb{Q}[X]$ de degré au plus n.

1. Donner un algorithme simple, reposant sur de l'algèbre linéaire, qui résout le problème DES_n et estimer le nombre d'opérations que cet algorithme utilise.
2. Dans le cas $k = 1$, le problème DES_n se spécialise au problème du calcul, à partir de deux polynômes $P, Q \in \mathbb{Q}[X]$ tels que $\deg(P) < \ell \cdot \deg(Q) = n$, des polynômes $C_1, \ldots, C_\ell$ vérifiant

$$\frac{P(X)}{Q^\ell(X)} = \sum_{j=1}^{\ell} \frac{C_j(X)}{Q^j(X)}, \quad \text{et } \deg(C_j) < \deg(Q) \text{ pour tout } j \geq 1.$$

Montrer que dans ce cas, le problème peut se résoudre en $O(M(n)\log n)$ opérations.

Le but des questions 3–9 ci-dessous est de montrer que le problème DES_n avec $\ell_i = 1$ pour tout i peut se résoudre en $O(M(n)\log n)$ opérations. Autrement dit, étant donnés les polynômes $P, R_1, \ldots, R_k$ dans $\mathbb{Q}[X]$, avec $\deg(P) < \sum_{i=1}^{k} \deg(R_i) = n$ et $R_1, \ldots, R_k$ deux à deux premiers entre eux, on veut calculer en $O(M(n)\log n)$ opérations les polynômes $C_1, \ldots, C_k \in \mathbb{Q}[X]$ tels que

$$\frac{P(X)}{R_1(X)\cdots R_k(X)} = \sum_{i=1}^{k} \frac{C_i(X)}{R_i(X)}, \quad \text{et } \deg(C_i) < \deg(R_i) \text{ pour tout } i \geq 1.$$

Pour ce faire, on introduit

$$R(X) = \sum_{i=1}^{k} \left(\prod_{\substack{j=1 \\ j\neq i}}^{k} R_j(X) \right)$$

et pour $1 \leq i \leq k$, on considère le reste $D_i(X)$ de la division euclidienne de $R(X)$ par $R_i(X)$.

3. Donner un algorithme qui calcule le polynôme $R(X)$ en $O(M(n)\log k)$ opérations.
4. Montrer que les polynômes $D_1, \ldots, D_k$ peuvent être calculés eux aussi en un nombre d'opérations en $O(M(n)\log k)$.
5. Montrer que pour tout $1 \leq i \leq k$, les polynômes D_i et R_i sont premiers entre eux.
6. En notant E_i l'inverse de D_i modulo R_i, montrer que C_i est égal au reste de la division euclidienne de $P \cdot E_i$ par R_i.
7. Donner un algorithme pour le calcul des E_i en $O(M(n)\log n)$ opérations.
8. Donner un algorithme pour le calcul des C_i, en $O(M(n)\log k)$ opérations.
9. Conclure qu'on peut résoudre le problème DES_n avec $\ell_i = 1$ pour tout i en coût $O(M(n)\log n)$.
10. Conclure qu'il est possible de résoudre le problème DES_n en $O(M(n)\log n)$ opérations.

■

Exercice 33.5 — Polynômes de Fibonacci. Soit $F_n(X) \in \mathbb{K}[X]$ la suite des polynômes de Fibonacci, définie par les conditions initiales $F_0 = 1$, $F_1 = X$ et la récurrence $F_{n+1} = XF_n + F_{n-1}$ pour tout $n \geq 1$. Soit $N \geq 1$.

1. Estimer le coût d'un calcul direct de tous les coefficients de $F_N(X)$.
2. Montrer qu'il est possible de calculer tous les coefficients de $F_N(X)$ en utilisant $O\big(M(N)\big)$ opérations dans $\mathbb{K}$.
3. Montrer qu'il est possible de calculer la somme des coefficients de F_N en seulement $O(\log N)$ opérations dans $\mathbb{K}$.
4. Donner un algorithme qui permet le calcul des coefficients de F_N en $O(N)$ opérations dans $\mathbb{K}$.

■

Exercice 33.6 — Fonctions algébriques et résultants. Soit $y(X)$ une fonction algébrique annulée par $P(X,Y) \in \mathbb{Q}[X,Y]$.

1. Montrer que la dérivée $y'(X)$ est aussi algébrique, en exhibant un polynôme résultant $R(X,Y)$ qui l'annule.
2. Déterminer un tel polynôme R dans le cas où $P(X,Y) = XY^2 - Y + 1$.
3. Supposant que P est de degré au plus d en X et en Y, estimer les degrés en X et en Y du polynôme R, puis proposer un algorithme pour son calcul.
4. Analyser la complexité de votre algorithme, mesurée en nombre d'opérations arithmétiques dans $\mathbb{Q}$. En utilisant les meilleurs algorithmes vus en cours pour les opérations fondamentales sur les polynômes et les matrices, à quel exposant aboutissez-vous ? Ici, par *exposant* on entend un réel α tel que votre algorithme a une complexité en $\tilde{O}(n^\alpha)$, c'est-à-dire linéaire en n^α à des facteurs logarithmiques près.

■

Exercice 33.7 — Produit diamant. On considère deux polynômes unitaires $f, g \in \mathbb{Q}[T]$, de degrés strictement plus petits que n. Étant donné $H \in \mathbb{Q}[X,Y]$, dont les degrés partiels par rapport à chacune des variables X et Y sont strictement inférieurs à n, le *produit diamant* $f \diamond_H g$ est le polynôme de degré $D = n^2$ défini par

$$f \diamond_H g = \prod_{f(\alpha)=0,\, g(\beta)=0} \Big(T - H(\alpha,\beta)\Big),$$

le produit étant pris sur les racines complexes de f et de g comptées avec multiplicités. On s'intéresse dans cet exercice au calcul efficace du produit diamant.

1. Montrer que le produit diamant peut s'exprimer à l'aide de résultants et que ses coefficients sont rationnels.

On note A l'algèbre quotient $\mathbb{Q}[X,Y]/(f(X), g(Y))$. Un élément $g \in A$ admet une écriture unique

$$g = \sum_{0 \le i,j < n} g_{i,j} x^i y^j$$

avec $g_{ij} \in \mathbb{Q}$, où x et y sont les images canoniques de X et Y dans A. On admet que si P_1, P_2 sont deux polynômes de $\mathbb{Q}[X,Y]$ de degrés au plus d_X en X et au plus d_Y en Y, on peut calculer leur produit $P_1 P_2$ en $O\big(M(d_X d_Y)\big)$ opérations dans $\mathbb{Q}$.

2. Montrer que $O\big(M(D)\big)$ opérations suffisent pour effectuer le produit de deux éléments de A.

À tout polynôme G de $\mathbb{Q}[X,Y]$ on associe l'application $\mathbb{Q}$-linéaire de multiplication par $G(x,y)$ dans A, définie par $v \in A \mapsto G(x,y)v \in A$. On admet que la trace, notée $\mathrm{Tr}(G)$, de cette application linéaire est égale à $\sum_{\alpha,\beta} G(\alpha,\beta)$, la somme étant prise sur les racines complexes α de f et β de g comptées avec multiplicités.

3. Montrer que, pour tout $s \in \mathbb{N}$, la trace $\mathrm{Tr}(H^s)$ s'exprime comme la somme des puissances s-ièmes des racines de $f \diamond_H g$.
4. Expliciter pour $0 \le i < n$ ce que valent $\mathrm{Tr}(X^i)$ et $\mathrm{Tr}(Y^i)$.

On rappelle que tout polynôme unitaire $P \in \mathbb{Q}[T]$ de degré au plus D peut être représenté soit par la suite de ces coefficients, soit par la suite des sommes des puissances s-ièmes de ses racines, pour $s = 0, \dots, D$, et qu'il est possible de passer d'une représentation à l'autre en $O\big(M(D)\big)$ opérations.

5. Donner un algorithme pour le calcul des valeurs $\mathrm{Tr}(X^i), 0 \le i < n$, en $O\big(M(n)\big) \subseteq O(D)$ opérations arithmétiques. Même question pour les valeurs $\mathrm{Tr}(Y^i)$, $0 \le i < n$.
6. En déduire que l'ensemble des valeurs $\mathrm{Tr}(X^i Y^j)$, pour $0 \le i, j < n$, peut être calculé en $O(D)$ opérations dans $\mathbb{Q}$.
7. Donner un algorithme qui, pour tout $N \ge 1$, calcule la suite

$$\mathrm{Tr}(1), \mathrm{Tr}(H), \mathrm{Tr}(H^2), \dots, \mathrm{Tr}(H^{N-1})$$

en $O\big(N\,M(D)\big)$ opérations dans $\mathbb{Q}$.

8. En déduire un algorithme pour le calcul de $f \diamond_H g$ en $O\big(D\,M(D)\big)$ opérations dans $\mathbb{Q}$.

■

Exercice 33.8 — Multiplication polynomiale de complexité quasi-linéaire. Le but de cet exercice est de montrer que la multiplication de polynômes de degré au plus N à coefficients dans un corps $\mathbb{K}$ contenant N éléments distincts en progression géométrique peut s'effectuer en complexité arithmétique quasi-linéaire par rapport à N.

Soit A et B deux polynômes de $\mathbb{K}[X]$, de degré strictement inférieurs à N. Soient n et d deux entiers tels que $N \le nd$ et écrivons

$$A = A_0(X) + A_1(X)X^d + \cdots + A_{n-1}(X)(X^d)^{n-1},$$

$$B = B_0(X) + B_1(X)X^d + \cdots + B_{n-1}(X)(X^d)^{n-1},$$

où les polynômes $A_i(X), B_i(X) \in \mathbb{K}[X]$ ont degré au plus $d-1$. Soit $\mathcal{G}$ un sous-ensemble de $\mathbb{K}$ contenant $2d$ éléments distincts en progression géométrique.

1. Si T opérations de $\mathbb{K}$ suffisent pour multiplier dans $\mathbb{K}[X, Y]$ les polynômes

$$A(X, Y) = A_0(X) + A_1(X)Y + \cdots + A_{n-1}(X)Y^{n-1} \quad \text{et}$$

$$B(X, Y) = B_0(X) + B_1(X)Y + \cdots + B_{n-1}(X)Y^{n-1},$$

montrer qu'on peut multiplier A(X) et B(X) en $T + O(nd)$ opérations de $\mathbb{K}$.

Pour calculer AB, il suffit donc d'effectuer le produit $C(X, Y) = A(X, Y)B(X, Y)$, c'est-à-dire de calculer les coefficients $C_i(X) \in \mathbb{K}[X]$, avec $\deg(C_i) < 2d$, de

$$C(X, Y) = C_0(X) + C_1(X)Y + \cdots + C_{2n-2}(X)Y^{2n-2}.$$

2. Montrer que le calcul du produit AB se ramène à $O(d)$ multiplications de $\mathbb{K}[Y]$ en degré $< n$ et $O(n)$ évaluations/interpolations sur les points de $\mathcal{G}$.
Indication : utiliser l'égalité $A(g, Y)B(g, Y) = C(g, Y)$, valable pour $g \in \mathcal{G}$.

3. En déduire qu'il existe trois constantes $\alpha, \beta, \gamma \geq 1$, universelles (c'est-à-dire indépendantes de N, n, d), telles que la complexité arithmétique M de l'algorithme de (2) vérifie $M(nd) \leq \alpha n M(d) + \beta d M(n) + \gamma nd$.
4. Écrire un algorithme récursif de multiplication dans $\mathbb{K}[X]$ de complexité arithmétique $O\left(N(\log N)^{\log(\alpha\beta)}\right)$ en degré N.

■

Exercice 33.9 — Multiplication d'opérateurs différentiels. Dans l'algèbre de polynômes non commutatifs $\mathbb{Q}[X]\langle\theta\rangle$ en les indéterminées X et $\theta = \frac{d}{dX}$ satisfaisant à la règle de commutation $\theta X = X\theta + X$, on considère deux éléments A et B donnés sous la forme

$$A = \sum_{i=0}^{n} \sum_{j=0}^{n} a_{i,j} X^i \theta^j \qquad \text{et} \qquad B = \sum_{i=0}^{n} \sum_{j=0}^{n} b_{i,j} X^i \theta^j$$

avec $a_{i,j} \in \mathbb{Q}$ et $b_{i,j} \in \mathbb{Q}$. Le but de cet exercice est d'étudier la complexité arithmétique du calcul du produit $C = BA$, c'est-à-dire du calcul, à partir des constantes $a_{i,j}, b_{i,j} \in \mathbb{Q}$, des constantes $c_{i,j} \in \mathbb{Q}$ telles que

$$C = \sum_{i=0}^{n} \sum_{j=0}^{n} c_{i,j} X^i \theta^j.$$

Les questions (1)–(3) qui suivent sont largement indépendantes.

1. a. Montrer que $\theta^i X^\ell = X^\ell (\theta + \ell)^i$ pour $i \geq 0$ et $\ell \geq 0$, et en déduire que le calcul du produit $\theta^i X^\ell$ peut s'effectuer en $O(i)$ opérations arithmétiques dans $\mathbb{Q}$.
   b. Donner des bornes, en fonction de n, sur les degrés de C en X et en θ.
   c. Proposer un algorithme naïf pour le calcul de C et estimer son coût arithmétique.
2. Montrer qu'il est possible de calculer C en $O\left(n^2 M(n)\right)$ opérations dans $\mathbb{Q}$, à l'aide d'un schéma itératif. On pensera à écrire B sous la forme

$$B = \sum_{i=0}^{n} b_i(X)\theta^i, \qquad \text{avec } b_i(X) \in \mathbb{Q}[X].$$

Dans la suite, on montre comment améliorer les résultats des questions (1) et (2), grâce à un algorithme de type évaluation-interpolation.

3. a. Calculer $\theta^j(X^k)$, pour $j \geq 0$ et $k \geq 0$. En déduire $P(\theta)(X^k)$, pour $k \geq 0$ et $P \in \mathbb{Q}[X]$.
   b. Montrer que les coefficients de C sont uniquement déterminés par la suite des polynômes $C(X^0), C(X^1), \ldots, C(X^{2n})$.
   c. Donner un algorithme de complexité arithmétique $O\left(n M(n) \log n\right)$ pour le calcul de tous les polynômes $p_k := A(X^k)$, pour $0 \leq k \leq 2n$, et $q_i := B(X^i)$, pour $0 \leq i \leq 3n$.

d. Montrer que le calcul des polynômes $\mathrm{B}(p_0),\dots,\mathrm{B}(p_{2n})$ se ramène à une multiplication de matrices de tailles $(4n+1)\times(3n+1)$ et $(3n+1)\times(2n+1)$.
e. Montrer que, à partir de $\mathrm{C}(\mathrm{X}^0),\dots,\mathrm{C}(\mathrm{X}^{2n})$, il est possible de calculer les coefficients de C en $\mathrm{O}\big(n\,\mathsf{M}(n)\log n\big)$ opérations arithmétiques.
f. Conclure en écrivant un algorithme complet pour le calcul de C basé sur (c), (d) et (e). Estimer sa complexité.

■

Exercice 33.10 — Une famille d'intégrales. L'intégrale

$$c_{n,k} = \int_0^{+\infty} t^k \mathrm{K}_0(t)^n \, dt,$$

où K_0 est une fonction de Bessel (voir ci-dessous), est intervenue récemment dans des études de physique liées au modèle d'Ising. Empiriquement, il est apparu que ces intégrales vérifient une récurrence de la forme

$$(k+1)^{n+1} c_{n,k} + \sum_{\substack{2\le j<n \\ j \text{ pair}}} \mathrm{P}_{n,j}(k) c_{n,k+j} = 0, \tag{R}$$

où les $\mathrm{P}_{n,j}$ sont des polynômes de degré au plus $n+1-j$.

Le but de cet exercice est de prouver l'existence d'une telle récurrence et de développer un algorithme permettant de la calculer. Le point de départ est l'équation différentielle

$$\theta^2(y) - t^2 y = 0$$

vérifiée par $y = \mathrm{K}_0(t)$. Dans cette équation, θ représente l'opérateur $t\frac{d}{dt}$. Ainsi, $\theta^2(y) = t(ty')'$. Il sera plus commode de travailler avec θ qu'avec d/dt, et on rappelle $\theta(uv) = u\theta(v) + \theta(u)v$.

On admettra que les intégrales $c_{n,k}$ convergent pour tous n et k entiers positifs, et que pour tout i, j entiers positifs on a en outre

$$\lim_{t\to 0} t^j (\mathrm{K}_0(t)^n)^{(i)} = \lim_{t\to\infty} t^j (\mathrm{K}_0(t)^n)^{(i)} = 0.$$

1. Montrer que K_0^2 vérifie l'équation différentielle

$$\theta^3(y) - 4t^2\theta(y) - 4t^2 y = 0.$$

2. Plus généralement, montrer que pour tout $n \in \mathbb{N}$, K_0^n vérifie une équation différentielle linéaire. Donner une borne supérieure sur son ordre. Vérifier que cette borne donne la bonne valeur pour $n = 2$.
3. En déduire l'existence d'une récurrence linéaire vérifiée par les $c_{n,k}$, pour chaque n. Donner un algorithme calculant cette récurrence.

4. Montrer par exemple que la suite $c_{2,k}$ vérifie

$$k^3 c_{2,k-1} = 4(k+1)c_{2,k+1}.$$

5. Borner le degré des coefficients dans l'équation calculée en question (2).
6. En déduire une borne de complexité sur l'algorithme de (3) en nombre d'opérations dans $\mathbb{Q}$.

La suite de l'exercice montre comment accélérer ce calcul en tirant parti du fait que l'équation différentielle vérifiée par K_0 est d'ordre 2 seulement.

7. Soit $\phi_k := n(n-1)\cdots(n-k+1)K_0^{n-k}\theta(K_0)^k$. Montrer que

$$\phi_{k+1}(t) = \theta(\phi_k)(t) - k(n-k+1)t^2\phi_{k-1}(t).$$

8. Utiliser cette récurrence pour $n = 2$ pour retrouver l'équation de la question (1), en récrivant $\phi_1, \phi_2, \ldots$ au moyen de ϕ_0.
9. Plus généralement, décrire un algorithme de calcul d'une équation différentielle linéaire vérifiée par K_0^n à partir du résultat de la question (7).
10. Montrer que la complexité du calcul de l'équation différentielle par cette méthode est bornée par $O(n^3)$.

■

Exercice 33.11 — Calcul rapide du polynôme caractéristique. On considère un réel $\theta > 2$ tel que la multiplication de deux matrices de $\mathcal{M}_n(\mathbb{K})$ peut se faire en $\mathsf{MM}(n) = n^\theta$ opérations dans $\mathbb{K}$. L'objectif de cet exercice est de prouver que le polynôme caractéristique d'une matrice « générique » de $\mathcal{M}_n(\mathbb{K})$ peut être calculé en $O(\mathsf{MM}(n))$ opérations de $\mathbb{K}$. Une matrice $F = (f_{ij})_{1\le i,j\le n}$ de $\mathcal{M}_n(\mathbb{K})$ sera appelée « de type m-Frobenius » si ses $n-m$ premières colonnes ne contiennent que des éléments nuls, à l'exception des éléments $f_{m+1,1}, f_{m+2,2}, \ldots, f_{n,n-m}$ qui valent tous 1. Par exemple, les matrices de type 1-Frobenius sont les matrices compagnon.

Dans la suite de l'exercice, on suppose que la taille n est une puissance de 2 et on note $n = 2^r$. On associe à A une suite de matrices $A_r = A, A_{r-1}, \ldots, A_1, A_0$ qui lui sont toutes semblables (c'est-à-dire de la forme $P^{-1}AP$, avec P inversible). Ces matrices sont définies de manière inductive. Pour passer de A_{i+1} à A_i, on calcule $s_i = n/2^i$ matrices, notées $A_{i,1} = A_{i+1}, A_{i,2}, \ldots, A_{i,s_i} = A_i$ et définies comme suit :

— On considère la matrice U_{ij} de type 2^i-Frobenius dont les 2^i dernières colonnes coïncident avec les 2^i dernières colonnes de A_{ij}.
On suppose que A est « suffisamment générique » pour que U_{ij} soit inversible.
— On pose $A_{i,j+1} := U_{ij}^{-1} A_{i,j} U_{ij}$.

On admettra sans preuve que la matrice A_i ainsi construite est de type 2^i-Frobenius.

1. Montrer que tous les éléments de l'inverse $D = (d_{ij})$ d'une matrice compagnon $C = (c_{ij})$ avec $c_{1n} \neq 0$ sont nuls, à l'exception des éléments suivants

$$d_{1,2} = \cdots = d_{n-1,n} = 1,\ d_{n,1} = {c_{1,n}}^{-1}, d_{n-j+1,1} = -c_{j,n}\, d_{n,1},\ \text{pour } 2 \le j \le n.$$

En déduire que les éléments de D peuvent se calculer en $O(n)$ opérations dans $\mathbb{K}$.
2. Étendre le résultat de la question (1), en montrant que l'inverse d'une matrice de type m-Frobenius peut se calculer en $O(n\mathsf{MM}(m)/m)$ opérations de $\mathbb{K}$.
3. En déduire que le passage de A_{i+1} à A_i se fait en $O(n^2\mathsf{MM}(2^i)/4^i)$ opérations de $\mathbb{K}$.
4. Conclure que le polynôme caractéristique de A peut se calculer en utilisant $O(\mathsf{MM}(n))$ opérations de $\mathbb{K}$.

■

Exercice 33.12 — Résolution de systèmes de Vandermonde transposés. Soient $a_0, a_1, \ldots, a_n$ des éléments distincts de $\mathbb{K}$ et soit V la matrice de Vandermonde $V = (a_i^j)_{0\le i,j\le n}$. Le but de cet exercice est de montrer que, pour tout vecteur $b \in \mathbb{K}^{n+1}$, on peut calculer l'unique solution $x \in \mathbb{K}^{n+1}$ du système linéaire ${}^tV \cdot x = b$ en $O(\mathsf{M}(n)\log n)$ opérations dans $\mathbb{K}$.

1. Donner un algorithme pour le calcul des coefficients du numérateur et du dénominateur de la fraction rationnelle $R = \sum_{i=0}^{n} \frac{1}{1-a_i X}$ en $O(\mathsf{M}(n)\log n)$ opérations dans $\mathbb{K}$.
2. Montrer qu'à partir de R, on peut calculer en $O(\mathsf{M}(n))$ opérations dans $\mathbb{K}$ les $2n$ premiers éléments de la suite $(z_i)_{i\ge 0}$ définie par $z_i = a_0^i + a_1^i + \cdots + a_n^i$.

On rappelle que, par définition, une matrice de type Hankel est constante le long de toutes ses anti-diagonales. On rappelle le théorème suivant vu au Chapitre 10 :

Si H *est une matrice de Hankel inversible de taille* $n+1$, *alors on peut résoudre le système linéaire* $Hu = b$ *en* $O(\mathsf{M}(n)\log n)$ *opérations dans* $\mathbb{K}$.

3. Montrer que la matrice $H := {}^tV \cdot V$ vérifie l'hypothèse de ce théorème.
4. Utiliser (1) et (2) pour estimer la complexité du calcul des éléments de H.
5. Conclure.
6. (Bonus) Écrire un algorithme résolvant ${}^tV \cdot x = b$, obtenu par transposition (au sens du Chapitre 12) de l'algorithme d'interpolation rapide vu au Chapitre 5.

■

Exercice 33.13 — Évaluation rapide de polynôme en une matrice. On rappelle que le polynôme caractéristique d'une matrice arbitraire de $\mathcal{M}_n(\mathbb{K})$ peut se calculer en $O(\mathsf{MM}(n)\log n)$ opérations de $\mathbb{K}$. Le but de cet exercice est d'étudier la complexité arithmétique du problème suivant :

> $\mathcal{E}(P, A)$: étant donnés un polynôme $P = p_0 + p_1 X + \cdots + p_D X^D$ de $\mathbb{K}[X]$ et une matrice $A \in \mathcal{M}_n(\mathbb{K})$, calculer la matrice $P(A) = p_0 I_n + p_1 A + \cdots + p_D A^D$.

1. Montrer que $O(\mathsf{MM}(n)\sqrt{D})$ opérations de $\mathbb{K}$ suffisent pour résoudre le problème $\mathcal{E}(P, A)$.
   Indication : écrire $P(X)$ sous la forme $P_0(X) + P_1(X)X^d + P_2(X)(X^d)^2 + \cdots$, avec d bien choisi et $P_i(X)$ de degrés au plus $d-1$.
2. Montrer qu'on peut améliorer ce résultat à $O(\mathsf{MM}(n)\sqrt{n} + \mathsf{M}(D))$ opérations.
   Indication : commencer par calculer le polynôme caractéristique de A.

3. Montrer qu'on peut résoudre $\mathcal{E}(\mathrm{X}^{\mathrm{D}}, \mathrm{A})$ en $\mathrm{O}(\mathsf{MM}(n)\sqrt{n} + \mathsf{M}(n)\log \mathrm{D})$ opérations de $\mathbb{K}$.
4. Montrer que si C est une matrice compagnon, alors il est possible de résoudre $\mathcal{E}(\mathrm{P}, \mathrm{C})$ en $\mathrm{O}(n^2 + \mathsf{M}(\mathrm{D}))$ opérations de $\mathbb{K}$.
5. Montrer que s'il existe un réel $\beta > 2$ tel que $\mathcal{E}(\mathrm{X}^2, \mathrm{A})$ puisse se résoudre en n^β opérations de $\mathbb{K}$ pour toute matrice A, alors la multiplication de deux matrices quelconques de $\mathcal{M}_n(\mathbb{K})$ peut se faire en $\mathrm{O}(n^\beta)$ opérations dans $\mathbb{K}$.
6. Montrer que si $\mathrm{A} \in \mathcal{M}_n(\mathbb{K})$ et $\deg(\mathrm{P}) = n$, la première colonne de P(A) peut être calculée en $\mathrm{O}(\mathsf{MM}(n)\log n)$ opérations de $\mathbb{K}$.

Dans la suite, on suppose que la matrice A est « suffisamment générique », en ce sens que la matrice $\mathrm{U} \in \mathcal{M}_n(\mathbb{K})$, dont la ℓ-ième colonne coïncide avec la première colonne de A^ℓ, est une matrice inversible.

7. Montrer que la matrice $\mathrm{C} = \mathrm{UAU}^{-1}$ est une matrice de type compagnon et qu'elle peut être calculée en $\mathrm{O}(\mathsf{MM}(n)\log n)$ opérations de $\mathbb{K}$.
   Indication : remarquer que l'espace vectoriel $\mathcal{V} = \mathbb{K}^n$ admet $\{\mathrm{A}^\ell e\}_{1 \le \ell \le n}$ comme base, où $e = {}^t(1, 0, \ldots, 0) \in \mathcal{V}$, et que U est la matrice de passage entre cette base et la base canonique de $\mathcal{V}$.
8. En déduire que, pour une matrice A « suffisamment générique », le problème $\mathcal{E}(\mathrm{P}, \mathrm{A})$ peut se résoudre en $\mathrm{O}(\mathsf{MM}(n)\log n + \mathsf{M}(\mathrm{D}))$ opérations de $\mathbb{K}$.
9. Montrer que, pour une matrice A « suffisamment générique », on peut résoudre $\mathcal{E}(\mathrm{X}^{\mathrm{D}}, \mathrm{A})$ en $\mathrm{O}(\mathsf{MM}(n)\log n + \mathsf{M}(n)\log \mathrm{D})$ opérations de $\mathbb{K}$.

■

Exercice 33.14 — Somme de solutions d'équations différentielles linéaires. On considère deux équations différentielles

$$a_r(\mathrm{X})y^{(r)}(\mathrm{X}) + \cdots + a_0(\mathrm{X})y(\mathrm{X}) = 0 \tag{E1}$$

et

$$b_s(\mathrm{X})y^{(s)}(\mathrm{X}) + \cdots + b_0(\mathrm{X})y(\mathrm{X}) = 0 \tag{E2}$$

et on recherche une équation différentielle vérifiée par toutes les sommes $h = f + g$ où f est solution de (E1) et g une solution de (E2). On suppose que les a_i et les b_i sont des polynômes de $\mathbb{Q}[\mathrm{X}]$ tels que $\deg a_i \le m$ et $\deg b_i \le n$, et qu'aucun de r, s, a_r et b_s n'est nul. On note $\mathrm{R} = r + s$ et $\mathrm{D} = \max\{m, n\}$.

Préambule

1. Montrer que h est solution d'une équation $\mathcal{R}(y) = 0$ d'ordre au plus R.

Les équations (E1) et (E2) permettent de récrire toute dérivée d'ordre arbitraire de f ou de g comme combinaison linéaire des $r + s$ fonctions $f, \ldots, f^{(r-1)}$, $g, \ldots$, et $g^{(s-1)}$. L'espace vectoriel sur $\mathbb{Q}(\mathrm{X})$ engendré par les dérivées d'ordre arbitraire $f, f', g'', \ldots, g, g', g'', \ldots$, a donc dimension au plus R.

Dans cet espace vectoriel, h, …, $h^{(\mathrm{R})}$ sont liées, et une relation de liaison non triviale fournit $\mathcal{R}$ après avoir chassé les dénominateurs.

Première méthode : recombinaison des équations

Pour $i \in \mathbb{N}$, $j \in \mathbb{N}$, $k \in \{1, 2\}$, on note $\mathcal{M}_{i,j}^{(k)}(y) = 0$ l'équation obtenue en dérivant j fois l'équation (k) puis en multipliant le résultat par X^i. On voit $\mathcal{M}_{i,j}^{(k)}$ comme une application de la variable y. Pour un entier $\delta \geq m + n$, on considère :

$$\mathrm{S}^{(\mathrm{E1})} = \left\{ \mathcal{M}_{i,j}^{(\mathrm{E1})} : i \leq \delta - m,\ j \leq s \right\} \quad \text{et} \quad \mathrm{S}^{(\mathrm{E2})} = \left\{ \mathcal{M}_{i,j}^{(\mathrm{E2})} : i \leq \delta - n,\ j \leq r \right\}.$$

2. Déterminer des bornes supérieures $d_{\max}$ et $r_{\max}$ telles que la famille des $\mathrm{X}^i y^{(j)}$ pour $0 \leq i \leq d_{\max}$ et $0 \leq j \leq r_{\max}$ permette d'exprimer tous les $\mathcal{R}(y)$ quand $\mathcal{R}$ décrit $\mathrm{S}^{(\mathrm{E1})} \cup \mathrm{S}^{(\mathrm{E2})}$.
3. On veut résoudre

$$\sum_{\mathcal{R} \in \mathrm{S}^{(\mathrm{E1})}} \lambda_{\mathcal{R}} \mathcal{R}(y) = \sum_{\mathcal{R} \in \mathrm{S}^{(\mathrm{E2})}} \mu_{\mathcal{R}} \mathcal{R}(y) \tag{R}$$

   en des coefficients λ et μ dans $\mathbb{Q}$. En égalant les coefficients sur la base des $\mathrm{X}^i y^{(j)}$, compter le nombre d'équations et le nombre d'inconnues. Choisir δ pour assurer l'existence de solutions non triviales.
4. Donner un algorithme pour le calcul d'une équation pour les sommes $h = f + g$ et justifier sa correction.
5. Évaluer en termes de R, D et avec des $\mathrm{O}(\cdot)$ le coût arithmétique du calcul des $\mathcal{M}_{i,j}^{(k)}(y)$ de proche en proche et celui de la résolution du système. Conclure sur la complexité arithmétique de la méthode.

Deuxième méthode : réductions par les équations

6. Pour une solution f de (E1) et une combinaison linéaire $u = p_0 f + \cdots + p_{r-1} f^{(r-1)}$ à coefficients des polynômes de degré maximal δ, montrer que $a_r u'$ s'exprime comme une combinaison linéaire de f, …, $f^{(r-1)}$ pour des coefficients polynomiaux dont on bornera le degré en fonction de δ.
7. En déduire que pour tout entier $i \geq r - 1$, le produit $a_r^{i+1-r} f^{(i)}$ s'exprime comme une combinaison linéaire $p_{i,0} f + \cdots + p_{i,r-1} f^{(r-1)}$ pour des polynômes $p_{i,j}$ dont on bornera le degré.
8. Énoncer un résultat analogue pour $b_s^{i+1-s} g^{(j)}$ et pour des polynômes $q_{i,j}$.
9. On veut résoudre

$$\sum_{i=0}^{\mathrm{R}} c_i(\mathrm{X}) f^{(i)} = 0 = \sum_{i=0}^{\mathrm{R}} c_i(\mathrm{X}) g^{(i)} \tag{P}$$

pour des polynômes c_i dans $\mathbb{Q}[X]$ communs aux deux équations. Montrer qu'en multipliant le membre de gauche de (P) par une puissance convenable de a_r et le membre de droite par une puissance convenable de b_s, et en réduisant par les équations (E1) et (E2), on obtient un système d'équations linéaires sur les c_i à coefficients polynomiaux. Indiquer les dimensions de ce système et borner le degré de ses coefficients en termes de R et D.

10. Donner un algorithme pour le calcul d'une équation pour les sommes $h = f + g$ et justifier sa correction.
11. Évaluer en termes de R, D et avec des $\tilde{O}(\cdot)$ (c'est-à-dire à des facteurs logarithmiques près) le coût arithmétique de cet algorithme.
12. Comparer avec la première méthode.

■

Exercice 33.15 — Quartique de Macaulay. On considère l'idéal homogène I engendré par les quatre polynômes

$$f_1 = X_3^3 - X_0^2X_2, \quad f_2 = X_2X_3 - X_0X_1,$$
$$f_3 = X_2^3 - X_1^2X_3, \quad f_4 = X_1X_3^2 - X_0X_2^2,$$

dans $\mathbb{Q}[X_0, X_1, X_2, X_3]$. On ordonne les monômes par l'ordre lexicographique induit par $X_3 < X_2 < X_1 < X_0$ et on convient de prendre comme monôme dominant d'un polynôme non nul le *plus petit* de ses monômes.

Pour un entier $n > 0$, on identifie librement un monôme $X_0^{e_0} \cdots X_n^{e_n}$ et le $(n+1)$-uplet d'entiers $(e_0, \ldots, e_n)$. Pour un idéal J de $\mathbb{Q}[X_0, \ldots, X_n]$, on note E(J) l'ensemble des monômes dominants des éléments non nuls de J, ou de façon équivalente, la partie stable correspondante dans $\mathbb{N}^{n+1}$.

1. Calculer une base standard de I relativement à l'ordre indiqué. Donner un système générateur minimal de $E(I) \subset \mathbb{N}^4$.
2. Rappelons que la fonction de Hilbert HF_E associée à une partie stable E de $\mathbb{N}^{n+1}$ est définie par :

$$HF_E(s) = \operatorname{card}\left\{(e_0, \ldots, e_n) \in \mathbb{N}^{n+1} \;\middle|\; (e_0, \ldots, e_n) \notin E,\ e_0 + \cdots + e_n = s\right\}.$$

   Pour i dans $\mathbb{N}$, on considère la partie de E(I) constituée des éléments dont la première coordonnée X_0 vaut i et on identifie cette partie à une partie E_i de $\mathbb{N}^3$.
   a. Donner un système générateur minimal de E_0 et dessiner E_0. Calculer la fonction de Hilbert, le polynôme de Hilbert, la dimension et le degré de E_0.
   b. Que vaut E_i pour un i général ?
   c. L'application induite par la multiplication par X_0 dans le quotient $\mathbb{Q}[X_0, X_1, X_2, X_3]/I$ est-elle injective ?
3. Calculer la fonction de Hilbert, le polynôme de Hilbert, la dimension et le degré de E(I).
4. Rappelons que l'espace projectif $\mathbb{P}^n_{\mathbb{C}}$ s'identifie à l'ensemble des $(n+1)$-uplets non nuls $(u_0 : \cdots : u_n)$ de $\mathbb{C}^{n+1}$ vus à multiplication près par un scalaire non nul.

On souhaite maintenant étudier le lieu des zéros Z(I) de l'idéal homogène I dans l'espace projectif $\mathbb{P}^3_{\mathbb{C}}$. Dans cette question, on considère X_1 comme indéterminée d'homogénéisation, et non comme d'habitude X_0.

a. Quel est l'ensemble des zéros à l'infini, c'est-à-dire de la forme $(x_0 : 0 : x_2 : x_3)$?
b. Déshomogénéiser le système polynomial initial en faisant $x_1 = 1$ et décrire l'ensemble algébrique affine de $\mathbb{C}^3$ ainsi obtenu.
c. Conclure en décrivant Z(I) et comparer avec les invariants calculés dans la question 3.

■

Exercice 33.16 — Diagonales de séries différentiellement finies. Étant donnée une série double différentiellement finie

$$f = \sum_{i,j\in\mathbb{N}} c_{i,j} x^i y^j, \tag{33.1}$$

cet exercice vise à donner un algorithme de calcul d'une équation différentielle pour la diagonale

$$\Delta(f) = \sum_{i\in\mathbb{N}} c_{i,i} t^i,$$

laquelle est aussi différentiellement finie. Par exemple, la série

$$\frac{1}{1-x-y} = \sum_{i,j\in\mathbb{N}} \binom{i+j}{i} x^i y^j, \tag{33.2}$$

qui vérifie deux équations différentielles linéaires d'ordre 1, l'une en x, l'autre en y, a pour diagonale

$$\Delta\left(\frac{1}{1-x-y}\right) = \sum_{i\in\mathbb{N}} \binom{2i}{i} t^i = \frac{1}{\sqrt{1-4t}}, \tag{33.3}$$

donnée par $(1-4t)y' - 2y = 0$.

En préambule, nous donnons quelques résultats à admettre et utiliser librement sur les deux natures de séries formelles un peu particulières utilisées dans cet énoncé. Ces résultats établissent que les séries considérées peuvent être manipulées formellement sans ambages (multiplications, dérivations, substitutions, identifications de termes).

Pour des coefficients $c_{i,j}$ dans un corps C, les séries du type (33.1) forment une C-algèbre notée C[[x, y]], stable par les dérivations par x et y, dont le corps des fractions $F_{x,y}$ contient celui des fractions rationnelles, C(x, y), et est lui-même stable par les deux dérivations. Une série f donnée par (33.1) étant aussi dans $F_{x,y}$, on dit qu'elle est différentiellement finie lorsque ses dérivées mixtes $\partial_x^i \partial_y^j(f)$ engendrent un sous-espace vectoriel de $F_{x,y}$ de dimension finie sur C(x, y).

Les séries

$$g = \sum_{i\in\mathbb{Z},\ j\in\mathbb{N},\ i+j\geq k} c_{i,j} s^i t^j,$$

où $k \in \mathbb{Z}$ ne dépend que de g, forment une autre C-algèbre, notée $S_{s,t}$, stable par les dérivations par s et t, dont le corps des fractions $F_{s,t}$ contient celui des fractions rationnelles, $C(s,t)$, et est lui-même stable par les deux dérivations. Une série g est dite différentiellement finie lorsque ses dérivées mixtes $\partial_s^i \partial_t^j(f)$ engendrent un sous-espace vectoriel de $F_{s,t}$ de dimension finie sur $C(s,t)$.

On admet que la substitution $x = t/s$, $y = s$ donne une application bien définie de $C[[x,y]]$ dans $S_{s,t}$.

1. On considère l'application C-linéaire ϕ de $C[[x,y]]$ dans $S_{s,t}$ définie sur f donnée par (33.1) par
$$\phi(f) = \frac{1}{s} f\left(\frac{t}{s}, s\right) = \sum_{i,j\in\mathbb{N}} c_{i,j} s^{j-i-1} t^i.$$
On note
$$\delta(f) = \sum_{i\in\mathbb{N}} c_{i,i} x^i y^i = \Delta(f)(xy).$$
Vérifier que
$$\phi\Big(\delta(f)\Big) = \frac{1}{s}\Delta(f).$$
2. Pour une série $f \in C[[x,y]]$, des fractions rationnelles $r_{i,j}(s,t)$ en nombre fini et un opérateur différentiel linéaire $L(t,\partial_t)$, on suppose vérifiée l'identité
$$L(t,\partial_t)(g) = \partial_s\left(\sum_{i,j} r_{i,j}(s,t)\partial_s^i\partial_t^j(g)\right)$$
quand $g = \phi(f)$. Montrer que L annule la diagonale $\Delta(f)$.
3. Pour $g = \phi(f)$, exprimer $\partial_s(g)$ et $\partial_t(g)$ comme les images par ϕ d'expressions linéaires en f et ses dérivées premières, qu'on exprimera sous la forme
$$\partial_s(g) = \phi\Big(L_s(f)\Big) \quad \text{et} \quad \partial_t(g) = \phi\Big(L_t(f)\Big).$$
Pour valider son résultat, on pourra observer que L_s et L_t commutent.
4. En déduire que si f est différentiellement finie, il en est de même de g.
5. Proposer un algorithme qui, étant donné un idéal annulateur pour f, calcule un idéal annulateur pour g.
6. En déduire un algorithme à base de la version différentielle de l'algorithme de Zeilberger qui, étant donné un idéal annulateur pour f, calcule un polynôme tordu $L(t,\partial_t)$ annulateur de $\Delta(f)$.
7. Montrer que $g = \phi(f) = 1/(s - t - s^2)$.
8. Appliquer à g l'algorithme de Zeilberger différentiel et produire une équation différentielle d'ordre 1 en t sur la diagonale (33.3).

■

Exercice 33.17 — Intégration définie de fractions rationnelles par réduction de Hermite. À l'exclusion de la première question, d'ordre plus général et indépendante de la suite, cet exercice s'intéresse à l'intégration définie rapide de fractions rationnelles à deux variables.

Pour la première question seulement, on considère l'intégration de fonctions f hyperexponentielles en deux variables x et y, c'est-à-dire dont les deux dérivées logarithmiques sont données par des fractions rationnelles :

$$\frac{\frac{\partial f}{\partial x}(x,y)}{f} \in \mathbb{Q}(x,y) \qquad \text{et} \qquad \frac{\frac{\partial f}{\partial y}(x,y)}{f} \in \mathbb{Q}(x,y).$$

1. Utiliser les algorithmes du cours pour décrire un analogue de l'algorithme de Zeilberger réalisant la création télescopique pour ce cadre, c'est-à-dire pour calculer un opérateur différentiel et une fraction rationnelle,

$$\mathrm{L} = \sum_{i=0}^{\rho} \eta_i(x)\partial_x^i \in \mathbb{Q}(x)\langle \partial_x; \mathrm{id}, d/dx\rangle \qquad \text{et} \qquad \phi \in \mathbb{Q}(x,y),$$

vérifiant la relation $\mathrm{L}(f) = \partial_y(\phi f)$.

Observons que cette approche ne permet pas de calculer L sans calculer ϕ, alors que ce dernier n'est pas toujours utilisé dans les applications, et par ailleurs que la taille de ce ϕ s'avère souvent être le facteur limitant dans les calculs.

On s'intéresse maintenant à l'intégration dans le cas de fonctions f qui soient des fractions rationnelles, afin d'obtenir un algorithme spécialisé de bonne complexité. Dorénavant, f est une fraction rationnelle de la forme p/q pour des polynômes non nuls p et q de $\mathbb{Q}[x,y]$ de degrés partiels en x et y bornés respectivement par d_x et d_y et tels que $\deg_y(p) < \deg_y(q)$. On suppose de plus que le polynôme $q \in \mathbb{Q}[x,y]$ est sans facteur carré (relativement à y), au sens où le pgcd de q et $\partial q/\partial y$, dans $\mathbb{Q}(x)[y]$, est 1. On rappelle qu'il existe alors une relation de Bézout de la forme

$$uq + v\frac{\partial q}{\partial_y} = 1$$

pour des polynômes à deux variables u et v dans $\mathbb{Q}(x)[y]$, vérifiant les contraintes $\deg_y(u) < d_y - 1$ et $\deg_y(v) < d_y$. La notation $\mathbb{Q}_{<d}[x]$, resp. $\mathbb{Q}_{\le d}[x]$, dénote l'ensemble des polynômes de degré strictement inférieur, resp. inférieur, à d. On admettra l'existence d'un algorithme qui, étant donnée une matrice $r \times r$ à coefficients polynomiaux de $\mathbb{Q}[x]$ de degré au plus n, calcule une base du noyau, constituée de vecteurs de polynômes, en $\tilde{\mathrm{O}}(r^\omega n)$ opérations dans $\mathbb{Q}$.

2. a. Pour $h \in \mathbb{Q}(x)[y]_{<2d_y}$, montrer l'existence de polynômes A et a dans $\mathbb{Q}(x)[y]_{<d_y}$ satisfaisant à

$$\frac{h}{q^2} = \partial_y\left(\frac{\mathrm{A}}{q}\right) + \frac{a}{q}.$$

b. Pour $\deg_y(h) < md_y$, généraliser ce résultat pour montrer l'existence de

polynômes $\mathrm{A} \in \mathbb{Q}(x)[y]_{<(m-1)d_y}$ et $a \in \mathbb{Q}(x)[y]_{<d_y}$ satisfaisant à

$$\frac{h}{q^m} = \partial_y\left(\frac{\mathrm{A}}{q^{m-1}}\right) + \frac{a}{q}.$$

On admettra l'unicité de a dans l'écriture précédente. Cette réduction de h/q^m à a/q est la réduction de Hermite.

3. On considère l'application ϕ qui à A et a dans $\mathbb{Q}(x)[y]_{<d_y}$ associe

$$\phi(\mathrm{A}, a) = q\frac{\partial \mathrm{A}}{\partial y} - \frac{\partial q}{\partial y}\mathrm{A} + qa$$

dans $\mathbb{Q}(x)[y]_{<2d_y}$, ainsi que sa matrice $\mathcal{H}$ sur la base monomiale. Expliciter les dimensions de cette matrice $\mathcal{H}$ ainsi qu'une borne sur le degré en x de ses coefficients.

4. On admettra que pour tout $h \in \mathbb{Q}[x, y]$ avec $\deg_y(h) < 2d_y$, l'équation $\phi(\mathrm{A}, a) = h$ admet une unique solution (A, a) pour des polynômes de $\mathbb{Q}(x)[y]$ vérifiant $\deg_y(\mathrm{A}) < d_y$ et $\deg_y(a) < d_y$. Ces polynômes peuvent donc se récrire avec un dénominateur commun δ sous la forme $\mathrm{A} = \mathrm{B}/\delta$ et $a = b/\delta$. On pose $\mu = \deg_x(h)$. Utiliser les formules de Cramer pour expliciter un bon choix de δ et donner des bornes sur les degrés en x de B, b et δ.

5. a. Donner un algorithme qui, étant donné un polynôme $h \in \mathbb{Q}[x, y]$ vérifiant $\deg_y(h) < 2d_y$, détermine B et b dans $\mathbb{Q}(x)[y]$ en fonction de h, u, v et q, de sorte que

$$\frac{h}{q^2} = \partial_y\left(\frac{\mathrm{B}}{\delta q}\right) + \frac{b}{\delta q},$$

avec $\deg_y(b) < d_y$.

b. Estimer la complexité de cet algorithme en supposant que B est renvoyé sans nécessairement développer les sommes et les produits, mais que b est renvoyé sous forme développée et réduite.

6. a. Par réduction de Hermite, chaque $\partial_x^i(f)$ peut s'écrire sous la forme

$$\partial_x^i(f) = \partial_y(g_i) + \frac{r_i}{q}$$

pour $g_i \in \mathbb{Q}(x, y)$ et un unique $r_i \in \mathbb{Q}(x)[y]$ tels que $\deg_y(r_i) < \deg_y(q)$. Donner les étapes d'un algorithme incrémental donnant r_{i+1} à partir de r_i et expliciter le g_{i+1} correspondant en fonction de g_i.

b. Donner un multiple explicite du dénominateur commun des coefficients de r_i et établir une borne sur le degré en x des numérateurs correspondants.

c. En déduire la complexité du calcul de tous les r_i pour $0 \le i \le \rho$.

7. a. Montrer que $\{r_0, \ldots, r_{d_y}\}$ est liée et en déduire un algorithme pour calculer un

opérateur différentiel et une fraction rationnelle,

$$\mathrm{L} = \sum_{i=0}^{\rho} \eta_i(x)\partial_x^i \in \mathbb{Q}(x)\langle \partial_x; \mathrm{id}, d/dx\rangle \qquad \text{et} \qquad g \in \mathbb{Q}(x,y),$$

vérifiant la relation $\mathrm{L}(f) = \partial_y(g)$. Décrire l'algorithme complet.

b. Évaluer sa complexité.

■

Exercice 33.18 — Rencontre entre une hyperbole et deux droites. Soit I l'idéal de $\mathbb{Q}[x,y]$ engendré par les deux polynômes $y^2 - x^2$ et $xy - 1$. On ordonne les monômes de $\mathbb{Q}[x,y]$ d'abord par degré croissant, puis en cas d'égalité de degré, par degré croissant en y :

$$1 < x < y < x^2 < xy < y^2 < x^3 < x^2y < xy^2 < y^3 < \dots$$

Le monôme dominant d'un polynôme non nul sera le plus grand pour cet ordre.

1. Calculer une base standard de I pour cet ordre et donner une base monomiale du $\mathbb{Q}$-espace vectoriel $\mathbb{Q}[x,y]/\mathrm{I}$.
2. La multiplication par x induit un endomorphisme linéaire de $\mathbb{Q}[x,y]/\mathrm{I}$. Calculer sa matrice dans la base ci-dessus, puis son polynôme caractéristique et ses valeurs propres. Que représentent ces dernières ?
3. Soit J un idéal de $\mathbb{Q}[x_1,\dots,x_n]$ définissant une variété affine V de $\mathbb{C}^n$ de dimension zéro, de degré r, et composée de r points distincts. Soit f un polynôme de $\mathbb{C}[x_1,\dots,x_n]$ séparant les points de V, c'est-à-dire que les images par f de deux points distincts de V restent distincts. On nomme P(T) le polynôme caractéristique de l'endomorphisme linéaire F de $\mathbb{Q}[x_1,\dots,x_n]/\mathrm{J}$ induit par la multiplication par f, et Q(T) le polynôme dont les racines sont les images par f des points de V.
   a. Quels sont les degrés de P et Q ?
   b. Soit x un point de V. Montrer que $f(x)$ est une valeur propre de F en construisant explicitement un vecteur propre par interpolation.

■

Exercice 33.19 — Problème des moments pour une suite hypergéométrique. Soit $(\mathrm{C}_n)_{n\geq 0} = (1,1,2,5,14,42,\dots)$ la suite de terme général $\mathrm{C}_n = \frac{1}{n+1}\binom{2n}{n}$. Le but de cet exercice est de résoudre le *problème des moments* pour cette suite, c'est-à-dire de trouver deux réels a et b, et une fonction continue $\mu : [a,b] \to \mathbb{R}_+$ tels que

$$\mathrm{C}_n = \int_a^b \mu(x)x^n\,dx \quad \text{pour tout } n \geq 0. \tag{P}$$

Une série algébrique. Soit $\mathrm{C}(z)$ la série génératrice $\mathrm{C}(z) = \sum_{n\geq 0} \mathrm{C}_n z^n$.

1. Expliciter une récurrence linéaire satisfaite par la suite (C_n).
2. Montrer que tout approximant de Padé-Hermite de type $(1,1,1)$ du vecteur de

séries ${}^t(1, C, C^2)$ est proportionnel à $(1, -1, z)$. En déduire un polynôme $P(z, w) \in \mathbb{Q}[z, w]$ tel que $P(z, C(z)) = 0 \bmod z^5$.
Pour cela, exprimer le problème d'approximation en termes d'algèbre linéaire. Expliciter la matrice du système linéaire associé. Est-elle structurée ? Quel est son rang de déplacement ?

3. Montrer qu'il existe une unique série $S(z) \in \mathbb{Q}[[z]]$ telle que $P(z, S(z)) = 0$. Combien vaut $S(z) \bmod z^5$?
4. Écrire un algorithme à base d'itérations de Newton, prenant en entrée un entier positif κ, et renvoyant en sortie les $N = 2^\kappa$ premiers coefficients de S.
5. Montrer que S vérifie une équation différentielle linéaire, que l'on explicitera. On exhibera d'abord une relation de Bézout dans $\mathbb{Q}(z)[w]$ entre P et $\frac{\partial P}{\partial w}$.
6. En déduire que les coefficients de $S(z) = \sum_{n\geq 0} s_n z^n$ vérifient la récurrence

$$(n+2)s_{n+1} - (4n+2)s_n = 0, \quad \text{pour tout } n \geq 0.$$

7. Conclure que la série $C(z)$ est algébrique, et est racine du polynôme $P(z, w)$.

Moments et résultants. Soit $G(z)$ la série $G(z) = C(1/z)/z$ dans $\mathbb{Q}[[1/z]]$.

Nous admettons que G définit une fonction sur $\mathbb{C} \setminus [0, 4]$, et que la solution du problème (P) est donnée par la *formule d'inversion de Stieltjes* :

$$\mu(x) = -\frac{\psi(x)}{\pi}, \quad \text{où} \quad \psi(x) = \lim_{y\to 0^+} \Big(\operatorname{Im} G(x + iy)\Big).$$

8. Montrer que $G(z)$ est une fonction algébrique, en exhibant un polynôme annulateur $K(z, w) \in \mathbb{Q}[z, w]$.
9. Écrivons $G(x + iy)$ sous la forme $A(x, y) + iB(x, y)$, avec A et B des fonctions réelles, et notons $\varphi(x)$ et $\psi(x)$ les limites en $y = 0^+$ de A et de B. Justifier l'égalité $K(x, \varphi(x) + i\psi(x)) = 0$. En déduire l'algébricité de $\varphi(x)$ et de $\psi(x)$.
10. Déterminer, à l'aide d'un résultant, un polynôme annulateur de $\psi(x)$. En déduire une forme explicite de $\mu(x)$.

Preuve par télescopage créatif. Nous allons prouver l'identité

$$C_n = \frac{1}{2\pi} \int_0^4 x^n \sqrt{\frac{4-x}{x}}\, dx. \tag{M}$$

11. On veut montrer que la fonction $U(n, x) = x^n \sqrt{\frac{4-x}{x}}$ est solution d'une équation fonctionnelle linéaire de la forme

$$(a(n)S_n + b(n)) \cdot U = D_x \cdot (r(n, x)U), \tag{C}$$

où S_n est l'opérateur de décalage, D_x est l'opérateur de dérivation, et où $r \in \mathbb{Q}(n, x)$ et $a, b \in \mathbb{Q}(n)$ sont des fractions rationnelles inconnues. (Ici, la notation « · » désigne l'application d'un opérateur à une fonction.)
Afin de déterminer a, b et r, on parcourra les étapes suivantes :

a. Expliciter deux opérateurs qui annulent U, de la forme $S_n - \alpha(n,x)$ et $D_x - \beta(n,x)$, avec $\alpha, \beta \in \mathbb{Q}(n,x)$.
b. Utiliser ces deux opérateurs pour montrer que l'équation (C) est équivalente à l'équation différentielle linéaire inhomogène paramétrée d'ordre 1

$$\frac{\partial r}{\partial x}(n,x) + \frac{nx - 4n + 2}{x(x-4)}\, r(n,x) = a(n)\,x + b(n).$$

c. Montrer que r n'a pas de pôle en $x = 0$ et $x = 4$. En déduire que r est un polynôme en x, à coefficients dans $\mathbb{Q}(n)$.
d. Borner le degré (en x) de r, et montrer que g est divisible par $x(x-4)$ dans $\mathbb{Q}(n)[x]$.
e. Expliciter g, puis a et b, et conclure.

12. En déduire une récurrence linéaire vérifiée par la suite $f_n = \int_0^4 U(n,x)\,dx$.
13. En admettant que $\displaystyle\int_0^4 \sqrt{\frac{4-x}{x}} = 2\pi$, conclure la preuve de l'égalité (M).

Compléments.

14. Quelle est la complexité arithmétique et binaire du calcul des N premiers termes de la série $S(z)$ en utilisant l'itération de Newton donnée en (4) ?
15. Quelle est la taille binaire de C_n, et quel est le coût binaire de son calcul, en utilisant la récurrence obtenue en (1) et un algorithme : (i) *naïf* ; (ii) par *scindage binaire* ; (iii) par « *pas de bébés, pas de géants* » ?

■

Exercice 33.20 — Relation de contiguïté entre fonctions de Bessel. Les fonctions de Bessel de première espèce, $J_n(x)$, sont solutions de l'équation différentielle et de l'équation mixte différentielle et de récurrence

$$x^2 y_n''(x) + x y_n'(x) + (x^2 - n^2) y_n(x) = 0 \quad \text{et} \quad x y_n'(x) + x y_{n+1}(x) - n y_n(x) = 0.$$

1. Écrire dans l'algèbre de Ore $\mathbb{Q}(n,x)\langle \partial_n, \partial_x; S_n, I, 0, d/dx\rangle$ les polynômes tordus associés à ces équations, et montrer qu'ils constituent une base de Gröbner pour l'ordre lexicographique induit par $\partial_n > \partial_x$.
2. Soit I l'idéal engendré par les polynômes tordus de la question précédente. Justifier, avec le moins de calculs possible, l'existence d'un polynôme de I non nul et indépendant de ∂_x. Quel est le degré minimal en ∂_n d'un tel polynôme ?
3. Indiquer une méthode de calcul de ce polynôme.

■

Exercice 33.21 — Équations différentielles pour les fonctions algébriques. On s'intéresse aux fonctions α solutions d'une équation polynomiale $P\big(x, \alpha(x)\big) = 0$ pour un polynôme $P(X,Y) \in \mathbb{Q}[X,Y]$ de degrés partiels D_X en X et D_Y en Y, avec $D_X \geq 1, D_Y > 1$. Nous voulons trouver des équations différentielles linéaires vérifiées par ces fonctions algébriques α.

On note P_X et P_Y les dérivées de P relativement à X et Y, respectivement. Pour éviter que P n'ait des facteurs multiples, on fait l'hypothèse que P et P_Y sont premiers entre eux.

Résolvante de Cockle. Un premier algorithme, attribué à Cockle (1861), représente les dérivées $\alpha^{(k)}$ comme des polynômes de degrés au plus $D_Y - 1$ en α et se ramène à de l'algèbre linéaire sur $\mathbb{Q}(X)$. Il passe par une première famille de polynômes W_k, introduits ci-dessous.

1. Montrer la relation $\alpha' = -P_X(x,\alpha)/P_Y(x,\alpha)$.
2. Montrer l'existence de polynômes $W_k(X,Y)$ tels que

$$\alpha^{(k)} = W_k(x,\alpha)/P_Y(x,\alpha)^{2k-1}$$

   en établissant une récurrence sur les polynômes W_k.
3. Que vaut W_1 ?
4. Montrer les inégalités $\deg_X W_k \leq (2D_X - 1)k - D_X$ et $\deg_Y W_k \leq 2(D_Y - 1)k - D_Y + 2$.

Cockle définit une deuxième suite $(V_k)_{k\geq 0}$ de polynômes de $\mathbb{Q}(X)[Y]$ de degrés au plus $D_Y - 1$ en Y, de sorte que, pour $k \geq 1$, $\alpha^{(k)} = W_k(x,\alpha)/P_Y(x,\alpha)^{2k-1} = V_k(x,\alpha)$, et par ailleurs, $V_0 = 1$.

5. Justifier que P_Y est inversible modulo P et indiquer comment se calcule V_1.
6. Donner un procédé de calcul itératif des V_k.
7. En déduire un algorithme par algèbre linéaire sur $\mathbb{Q}(X)$ pour calculer une équation différentielle linéaire d'ordre minimal annulant toute solution $\alpha(x)$ de $P\big(x,\alpha(x)\big) = 0$.
8. Soit r l'ordre de la sortie de cet algorithme. Donner une borne simple sur cet ordre, valable pour tout P.

On s'intéresse à la complexité de cette approche en nombre d'opérations dans $\mathbb{Q}$ et on commence par donner des bornes sur les degrés des résultats intermédiaires. Tout polynôme Q de $\mathbb{Q}(X)[Y]$ peut s'écrire sous la forme $q(X)^{-1}\,\bar{Q}(X,Y)$ pour $\bar{Q}$ dans $\mathbb{Q}[X,Y]$. On appellera degré de Q en X le maximum des degrés en X de q et $\bar{Q}$. On note $p(X)$ le coefficient dominant de P.

On admettra l'existence de polynômes A et B de $\mathbb{Q}[X,Y]$ tels que

$$\mathrm{Res}_Y(P,P_Y) = AP + BP_Y,$$

et tels que les degrés en Y de A et B sont dans $O(D_Y)$ et les degrés en X sont dans $O(D_X D_Y)$.

9. Pour un polynôme $Q(X,Y)$ de $\mathbb{Q}[X,Y]$, de degrés respectifs δ_X et δ_Y en X et Y, on considère sa division euclidienne par P sous la forme $p^{\delta_Y - D_Y}Q = UP + V$, pour U et V dans $\mathbb{Q}(X)[Y]$, avec V de degré en Y inférieur à δ_Y. Expliquer que U et V sont en réalité des polynômes de $\mathbb{Q}[X,Y]$. Montrer que le reste V vérifie

$$\deg_X V \leq \delta_X + (\delta_Y - D_Y)D_X.$$

10. Pour un polynôme $Q(X,Y)$ de $\mathbb{Q}(X)[Y]$, de degrés respectifs δ_X et δ_Y en X et Y, expliciter une borne sur le degré en X du reste de Q modulo P.

11. Déduire du calcul des V_k que le degré en X de V_k est dans $O(kD_XD_Y)$ pour tout $k \geq 1$.

On passe maintenant à l'étude de complexité proprement dite. On pourra se contenter de donner des bornes sous la forme de $\tilde{O}(\,)$ de quantités polynomiales en D_X, D_Y et k.

12. Donner en la justifiant la complexité du calcul de V_{k+1} à partir de V_k.
13. Décrire les systèmes linéaires considérés à la question 7 : quelles sont leurs tailles et les degrés de leurs coefficients ? Donner et justifier la complexité de leur résolution.
14. Conclure en donnant une borne de complexité pour l'algorithme de Cockle. Quelle est la taille arithmétique totale de sa sortie ?

Par conception, l'algorithme de Cockle recherche une équation d'ordre minimal. En relâchant cette contrainte sur l'ordre, on peut grandement abaisser la taille arithmétique de la sortie.

Approche par télescopage créatif. On note $F = YP_Y/P$ et on admet la formule suivante, qui voit α comme un résidu :

$$\alpha(x) = \frac{1}{2i\pi} \oint F(x, y)\, dy$$

pour un contour d'intégration qui tourne autour d'$\alpha(x)$ sans contenir aucun autre zéro de $P(x, Y)$.

On rappelle que le résidu d'une fraction rationnelle $S(Y)$ en un de ses pôles y d'ordre ρ est le coefficient c_{-1} de $1/(Y-y)$ dans le développement

$$S(Y) = \frac{c_{-\rho}}{(Y-y)^\rho} + \cdots + \frac{c_{-1}}{Y-y} + \sum_{n\geq 0} c_n \cdot (Y-y)^n.$$

On considère l'algèbre de Ore $\mathbb{Q}(X, Y)\langle \partial_X, \partial_Y; \mathrm{id}, \mathrm{id}, d/dX, d/dY\rangle$ que l'on fait agir sur les fractions rationnelles. Soit

$$\Lambda(X, Y, \partial_X, \partial_Y) = A(X, \partial_X) + \partial_Y\, B(X, Y, \partial_X, \partial_Y)$$

un polynôme tordu.

1. Que vaut le résidu de $\partial_Y(S)$ en l'un quelconque de pôles y de S ?
2. Si Λ annule une fraction rationnelle S, montrer que A annule tout résidu de S en l'un quelconque de ses pôles y.

On veut maintenant construire un polynôme tordu Λ annulant F par recombinaison linéaire.

3. Montrer que les fractions rationnelles

$$F_{i,j,k} = X^i \partial_X^j \partial_Y^k(F), \qquad 0 \leq i \leq N,\ 0 \leq j+k \leq M$$

s'écrivent comme combinaisons linéaires sur $\mathbb{Q}$ des éléments

$$\mathrm{E}_{i,j} = \frac{\mathrm{X}^i \mathrm{Y}^j}{\mathrm{P}^{\mathrm{M}+1}}, \qquad 0 \le i \le \mathrm{N} + \mathrm{D}_\mathrm{X}(\mathrm{M}+1),\ 0 \le j \le \mathrm{D}_\mathrm{Y}(\mathrm{M}+1).$$

4. Pour des entiers N et M fixés, combien y a-t-il de $\mathrm{F}_{i,j,k}$ et combien y a-t-il de $\mathrm{E}_{i,j}$?
5. Fixer $\mathrm{N} = 3\mathrm{D}_\mathrm{X}\mathrm{D}_\mathrm{Y} - 1$ et $\mathrm{M} = 6\mathrm{D}_\mathrm{Y} - 1$ et montrer l'existence de Λ non nul qui annule F.

On s'intéresse maintenant à la complexité du calcul de Λ.

6. Décrire un calcul de l'ensemble des $\mathrm{F}_{i,j,k}$ exprimés en termes des $\mathrm{E}_{i,j}$ et en donner la complexité.
7. Indiquer comment en déduire Λ et donner la complexité de cette étape.
8. Quelle est la complexité globale du calcul de Λ ? Quelle est sa taille arithmétique totale ?

Dans les bons cas, on trouve $\mathrm{A} \neq 0$ qui annule α ; quand A est nul, on peut montrer l'existence d'un autre annulateur de α dont les degrés ne sont pas plus que le double de ceux de A. ∎

Exercice 33.22 — Transporteur de deux idéaux. Soit $\mathbb{K}$ un corps. Soient I et J deux idéaux de $\mathrm{R} := \mathbb{K}[x_1, \ldots, x_n]$. On définit le *transporteur* de J dans I, noté $(\mathrm{I} : \mathrm{J})$, comme $(\mathrm{I} : \mathrm{J}) := \{f \in \mathrm{R} \mid f\mathrm{J} \subseteq \mathrm{I}\}$.

1. Montrer que $(\mathrm{I} : \mathrm{J})$ est un idéal.
2. Montrer que pour tout idéal K de R l'inclusion $\mathrm{KJ} \subseteq \mathrm{I}$ est équivalente à $\mathrm{K} \subseteq (\mathrm{I} : \mathrm{J})$.
3. Montrer que si $\mathrm{J} \subseteq \mathrm{I}$ alors $(\mathrm{I} : \mathrm{J}) = \mathrm{R}$.
4. Montrer que $(\mathrm{I} : \mathrm{R}) = \mathrm{I}$.
5. Soient J_1 et J_2 deux idéaux de R. Montrer que $(\mathrm{I} : (\mathrm{J}_1 + \mathrm{J}_2)) = (\mathrm{I} : \mathrm{J}_1) \cap (\mathrm{I} : \mathrm{J}_2)$.
6. Montrer que $(\mathrm{I} : (r)) = \frac{1}{r}(\mathrm{I} \cap (r))$, où $\frac{1}{r}(\mathrm{I} \cap (r))$ représente $\{\frac{f}{r} \mid f \in \mathrm{I} \cap (r)\}$.
7. Proposer un algorithme, utilisant un calcul de base standard, pour calculer $\mathrm{I} \cap \mathrm{J}$ lorsque I et J sont donnés par un ensemble de polynômes générateurs $\mathrm{I} = (f_1, \ldots, f_s)$ et $\mathrm{J} = (g_1, \ldots, g_r)$. On pourra, par exemple, considérer l'idéal $\mathrm{K} := (tf_1, \ldots, tf_s, (1-t)g_1, \ldots, (1-t)g_r)$ de $\mathbb{K}[x_1, \ldots, x_s, t]$, où t est une nouvelle variable.
8. Lorsque I est donné par un ensemble de polynômes générateurs $\mathrm{I} = (f_1, \ldots, f_s)$, en déduire un algorithme pour calculer un ensemble de polynômes générateurs de $(\mathrm{I} : (r))$.
9. Supposons que I et J sont explicitement donnés par des polynômes générateurs $\mathrm{I} = (f_1, \ldots, f_s)$ et $\mathrm{J} = (g_1, \ldots, g_r)$. À partir des résultats précédents, proposer un algorithme pour calculer $(\mathrm{I} : \mathrm{J})$.
10. Si $\mathbb{K}$ est algébriquement clos, et si V et W sont deux variétés affines de $\mathbb{A}^n_\mathbb{K}$, alors montrer que $(\mathbf{I}(\mathrm{V}) : \mathbf{I}(\mathrm{W})) = \mathbf{I}(\mathrm{V} \setminus \mathrm{W})$ (on rappelle que $\mathbf{I}(\mathrm{V})$ représente l'ensemble des polynômes de R qui s'annulent sur V).

∎

Exercice 33.23 On s'intéresse à la somme $\mathrm{F}_n := \sum_{k=0}^{n} k\binom{n}{k}$, pour laquelle on veut trouver une forme explicite. On pose $f_{n,k} := k\binom{n}{k}$.

1. Calculer $f_{n+1,k}$, $f_{n,k+1}$ et $f_{n+1,k+1}$.
2. Montrer que la combinaison linéaire
$$\mathrm{A}f_{n,k} + \mathrm{B}f_{n+1,k} + \mathrm{C}f_{n,k+1} + \mathrm{D}f_{n+1,k+1}$$
s'annule si et seulement si un certain polynôme s'annule.
3. Déterminer des constantes A, B, C et D qui annulent ce polynôme.
4. En déduire, par télescopage créatif qu'il conviendra de justifier, que la somme vérifie la récurrence
$$n\mathrm{F}_{n+1} = 2(n+1)\mathrm{F}_n.$$
5. En déduire une forme explicite pour F_n.
6. Proposer une méthode de sommation pour des suites hypergéométriques qui généralise cet exemple. On tâchera de fournir une formule explicite pour le télescopage créatif.

■

Exercice 33.24 On s'intéresse à la somme $\mathrm{F}_n := \sum_{k=0}^{n} \binom{n}{2k}\binom{2k}{k}\frac{1}{4^k}$, pour laquelle on souhaite vérifier par l'algorithme de Zeilberger qu'elle vaut $\frac{1}{2^{n-1}}\binom{2n-1}{n-1}$. On pourra suivre les indications données.

1. Représenter $f_{n,k} := \binom{n}{2k}\binom{2k}{k}\frac{1}{4^k}$ à l'aide de factorielles et calculer $f_{n+1,k}$ et $f_{n,k+1}$.
2. Passer directement à l'étape de l'algorithme de Zeilberger qui recherche une récurrence en n d'ordre 1. Écrire explicitement la récurrence qui relie une combinaison linéaire des décalées en n de f et une différence finie en k du terme $\mathrm{R}(n,k)f_{n,k}$. Quelles sont les inconnues de ce problème ? Dans quels ensembles cherche-t-on les solutions ?

Dans ce qui suit, on ne notera plus la dépendance de R en n et on écrira simplement $\mathrm{R}(k)$, $\mathrm{R}(k+1)$, etc.

3. Justifier que le seul dénominateur possible de R est $n-2k+1$. Écrire R sous la forme $\mathrm{S}(k)/(n-2k+1)$ pour un polynôme S en k (à coefficient des fractions rationnelles en n) et remplacer dans l'équation.
4. On se restreint momentanément à l'équation homogène associée, c'est-à-dire au membre qui fait apparaître S. Remplacer S par un polynôme unitaire de degré α en k et développer juste suffisamment pour justifier que l'équation homogène ne peut avoir de solution polynomiale.
5. En déduire que le degré des solutions polynomiales de l'équation complète est 2.
6. Poser $\mathrm{S}(k) = \mathrm{A}k^2 + \mathrm{B}k + \mathrm{C}$. Relier d'abord η et A.

7. Spécialiser en $k = -1$ pour obtenir une expression très simple pour C.
8. Spécialiser en $k = 0$ pour relier A et B.
9. Spécialiser en $k = (n+1)/2$ pour obtenir une autre relation entre A et B.
10. Conclure sur les valeurs de A, B, C et η.
11. En déduire que la somme F_n vérifie $(n+1)\mathrm{F}_{n+1} = (2n+1)\mathrm{F}_n$.
12. Vérifier que la forme explicite annoncée est solution et prouver qu'elle est bien forme explicite de la somme des $f_{n,k}$.

■

Exercice 33.25 — Composition itérée de séries formelles. Soit F une série de la forme $\mathrm{F}(\mathrm{X}) = f_1\mathrm{X} + f_2\mathrm{X}^2 + \cdots$ à coefficients dans un corps $\mathbb{K}$. En notant $\mathrm{F}^{[0]}(\mathrm{X}) = \mathrm{X}$, on définit la q-ième itérée de F par

$$\mathrm{F}^{[q]}(\mathrm{X}) = \mathrm{F}^{[q-1]}(\mathrm{F}(\mathrm{X})). \tag{E}$$

L'objectif de cet exercice est d'étudier la complexité $\mathrm{C}_q(\mathrm{N})$ du calcul des N premiers termes de cette série pour N grand en nombre d'opérations dans $\mathbb{K}$. On note $\mathsf{M}(\mathrm{N})$ une borne sur le nombre d'opérations dans $\mathbb{K}$ nécessaires pour calculer le produit de deux polynômes de degré N dans $\mathbb{K}[\mathrm{X}]$. On fait sur M les hypothèses habituelles de régularité. On note également $\mathrm{C}(\mathrm{N})$ une borne sur le nombre d'opérations dans $\mathbb{K}$ nécessaires pour calculer la composition $f(g(\mathrm{X}))$ de deux séries tronquées à l'ordre N telles que $g(0) = 0$. On rappelle qu'aucun algorithme quasi-optimal n'est connu pour la composition et que la meilleure complexité connue est $\mathrm{C}(\mathrm{N}) = \mathrm{O}(\sqrt{\mathrm{N}\log\mathrm{N}}\mathsf{M}(\mathrm{N}))$.

La notation $g(q,\mathrm{N}) = \mathrm{O}(f(q,\mathrm{N}))$ pour des fonctions de q et N voudra dire dans cet exercice qu'il existe un entier N_0 tel que pour tout $\mathrm{N} > \mathrm{N}_0$,

$$|g(q,\mathrm{N})| \leq \mathrm{K}|f(q,\mathrm{N})|,$$

la constante K ne dépendant ni de N, ni de q.

1. En n'utilisant que de l'algorithmique naïve, montrer que le calcul des N premiers termes de $\mathrm{F}^{[q]}$ peut être effectué en $\mathrm{C}_q(\mathrm{N}) = \mathrm{O}(q\mathrm{N}^3)$ opérations dans $\mathbb{K}$.
2. En exploitant la composition rapide de séries, montrer que cette complexité peut être abaissée à $\mathrm{C}_q(\mathrm{N}) = \mathrm{O}(q\sqrt{\mathrm{N}\log\mathrm{N}}\mathsf{M}(\mathrm{N}))$.
3. Décrire une idée simple permettant de réduire la dépendance en q pour aboutir à $\mathrm{C}_q(\mathrm{N}) = \mathrm{O}(\log q\sqrt{\mathrm{N}\log\mathrm{N}}\mathsf{M}(\mathrm{N}))$.

Le reste de l'exercice étudie un algorithme permettant d'abaisser encore cette complexité pour finir à la même complexité que la composition avec F, indépendamment de q. L'idée de départ est que dans le cas de la puissance, le calcul de $\mathrm{F}(\mathrm{X})^q$ peut être réalisé efficacement sous la forme $\exp(q\log\mathrm{F}(\mathrm{X}))$. L'objectif est d'adapter cette technique à la composition.

On suppose maintenant que $f_1 \neq 0$ et que f_1 n'est pas une racine de l'unité.

On définit la série de Schroeder $\mathrm{S}(\mathrm{X}) = s_1\mathrm{X} + s_2\mathrm{X}^2 + \cdots$ par

$$\mathrm{S}(\mathrm{F}(\mathrm{X})) = f_1\mathrm{S}(\mathrm{X}), \qquad s_1 = 1. \tag{S}$$

4. Montrer que cette équation définit bien une série S(X), et que celle-ci est unique.
5. Montrer que pour tout entier q,

$$F^{[q]}(X) = S^{[-1]}(f_1^q S(X)),$$

où $S^{[-1]}$ est l'inverse de S pour la composition. En déduire une borne sur la complexité du calcul de $F^{[q]}$ une fois la série S connue à précision N.
6. Pour calculer S solution de (S), on va s'intéresser à la résolution d'une équation plus générale :

$$A(X)Y(F(X)) - B(X)Y(X) - C(X) = 0 \bmod X^k, \tag{E}$$

où A, B, C, F sont des séries données et Y est l'inconnue. L'approche est une technique de « diviser pour régner ». En posant

$$Y(X) = U(X) + X^n V(X),$$

où U est un polynôme de degré inférieur à n, montrer que U et V sont solutions d'équations du même type que (E). Expliciter les coefficients de ces équations.
7. Décrire un algorithme de calcul des N premiers coefficients de Y en nombre d'opérations $O(\sqrt{N \log N} M(N))$. En notant a_0, b_0, c_0 les termes constants des séries A, B, C, on supposera que $a_0 f_1^m \neq b_0$, pour $m = 1, 2, 3, \ldots$, et que si $a_0 = b_0$, alors $c_0 = 0$.
8. Décrire enfin l'algorithme de calcul de $F^{[q]}$ dans la complexité annoncée.
9. À l'inverse, décrire un algorithme calculant une série G telle que $F = G^{[q]}$, dans la même complexité.

■

Exercice 33.26 — Calcul du radical. On rappelle que le radical d'un idéal I, noté $\sqrt{I}$, est défini par $\{f \mid \exists r \in \mathbb{N}, f^r \in I\}$. On rappelle aussi qu'un polynôme est dit sans carré si les multiplicités de ses facteurs irréductibles sont égales à un. Dans cet exercice $\mathbb{K}$ représente un corps de caractéristique zéro, et R est l'anneau des polynômes $\mathbb{K}[X_1, \ldots, X_n]$.

1. Si $n = 1$, et si I est un idéal de R engendré par $f_1, \ldots, f_s$, décrire un algorithme pour calculer un ensemble de générateurs du radical de I.
2. Dans cette question nous supposons que, pour chaque $i \in \{1, \ldots, n\}$, il existe $g_i \in I \cap \mathbb{K}[X_i]$ sans carré, et souhaitons montrer que I est radical, en suivant les étapes suivantes :
   a. Si $n = 1$ montrer que I est radical.
   b. Si $h_1, \ldots, h_m$ sont les facteurs irréductibles de g_1 alors montrer l'égalité $I = \bigcap_{j=1}^m (I + (h_j))$, en utilisant le fait qu'il existe des polynômes $v_1, \ldots, v_m$ de $\mathbb{K}[X_1]$ tels que $v_1 \frac{g_1}{h_1} + \cdots + v_m \frac{g_1}{h_m} = 1$.
   c. Montrer que si $I + (h_j)$ est radical pour tout $j \in \{1, \ldots, m\}$, alors I est radical.
   d. Soit $\mathbb{L}$ l'extension du corps $\mathbb{K}$ définie par $\mathbb{K}[X_1]/(h_1(X_1))$ et soit ϕ la projection $\mathbb{K}[X_1, \ldots, X_n] \to \mathbb{L}[X_2, \ldots, X_n]$. Montrer que si $J := \phi(I + (h_1))$ est radical

alors $I + (h_1)$ est radical.

e. Finalement, en utilisant les questions précédentes, montrer par récurrence sur n que I est radical.

3. On rappelle que I est dit de dimension affine zéro si la dimension de la $\mathbb{K}$-algèbre R/I est finie. Nous supposons dans cette question que I est un idéal de dimension affine zéro engendré par les polynômes $f_1, \ldots, f_s$, et nous nous intéressons au calcul d'un ensemble de générateurs de son radical.
   a. Montrer que pour tout $i \in \{1, \ldots, n\}$, il existe un polynôme sans carré g_i dans $\sqrt{I} \cap \mathbb{K}[X_i]$ et décrire un algorithme pour calculer de tels polynômes g_i.
   b. Montrer que $\sqrt{I}$ est engendré par $f_1, \ldots, f_s, g_1, \ldots, g_n$.

■

Exercice 33.27 — Terminaison de l'algorithme de Zeilbergersur la classe des termes proprement hypergéométriques. Cet exercice montre la terminaison de l'algorithme de Zeilberger sur la classe des *termes proprement hypergéométriques*, qui sont des termes hypergéométriques de la forme

$$h_{n,k} = P(n,k)\,\zeta^n X i^k \prod_{\ell=1}^{L} \Gamma(a_\ell n + b_\ell k + c_\ell)^{\epsilon_\ell}, \tag{33.4}$$

où, pour un corps $\mathbb{K}$ inclus dans $\mathbb{C}$, les a_ℓ et les b_ℓ sont dans $\mathbb{Z}$, les ϵ_ℓ valent ± 1, les c_ℓ, ζ et Xi sont dans $\mathbb{K}$, et P est un polynôme dans $\mathbb{K}[n,k]$. On rappelle que la fonction gamma vérifie pour $s \in \mathbb{C} \setminus \mathbb{Z}^-$, $\Gamma(s+1) = s\Gamma(s)$, et pour tout $n \in \mathbb{N}$, $\Gamma(n+1) = n!$.

1. Rappeler comment l'existence d'une relation

$$\sum_{i=0}^{r} \sum_{j=0}^{s} f_{i,j}(n)\, h_{n+i,k+j} = 0 \tag{33.5}$$

pour $r, s \geq 0$ et des fractions rationnelles $f_{i,j} \in \mathbb{K}(n)$ non toutes nulles entraîne l'existence de fractions rationnelles $\theta_0, \ldots, \theta_r \in \mathbb{K}(n)$ et $q \in \mathbb{K}(n,k)$, non toutes nulles, vérifiant la relation

$$\theta_r(n)\, h_{n+r,k} + \cdots + \theta_0(n)\, h_{n,k} = q(n,k+1)\, h_{n,k+1} - q(n,k)\, h_{n,k}. \tag{33.6}$$

2. Donner un exemple de terme hypergéométrique qui ne peut pas s'exprimer comme terme proprement hypergéométrique.
3. Pour chaque ℓ, borner en fonction de r et s la valeur absolue de l'entier u intervenant dans les termes $\Gamma(a_\ell n + b_\ell k + c_\ell + u)$ apparaissant dans l'égalité (33.5) après prise en compte de la définition (33.4) et des décalages sur n et k.
4. Montrer que, pour les entiers u définis comme à la question précédente, les deux quotients

$$\frac{\Gamma(a_\ell n + b_\ell k + c_\ell + u)}{\Gamma(a_\ell n + b_\ell k + c_\ell - \sigma_\ell)} \quad \text{et} \quad \frac{\Gamma(a_\ell n + b_\ell k + c_\ell + \sigma_\ell)}{\Gamma(a_\ell n + b_\ell k + c_\ell + u)}$$

sont des polynômes dont on majorera le degré en fonction de la borne σ_ℓ obtenue à la question précédente.

5. Construire un terme $\mathrm{H}_{n,k}$ tel que chaque produit $\mathrm{H}_{n,k}h_{n+i,k+j}$ pour $0 \le i \le r$ et $0 \le j \le s$ soit un polynôme de degré en k au plus linéaire en r et s.
6. En déduire que pour i et j comme à la question précédente

$$\deg_k\Big(\mathrm{H}_{n,k}h_{n+i,k+j}\Big) \le \deg_k(\mathrm{P}) + \mathrm{A}r + \mathrm{B}s$$

pour des entiers A et B que l'on précisera.

7. En déduire une contrainte polynomiale sur r et s pour que l'équation (33.5) ait une solution non triviale $\{f_{i,j}\}_{0\le i\le r,\ 0\le j\le s}$ pour des $f_{i,j}$ dans $\mathbb{K}(n)$.
8. En déduire que le choix

$$r = \mathrm{B} \quad \text{et} \quad s = (\mathrm{A}-1)\,\mathrm{B} + \deg_k(\mathrm{P}) + 1$$

fournit toujours une solution à l'équation (33.5).

9. Décrire à partir de l'analyse précédente un algorithme prenant en entrée un terme proprement hypergéométrique de la forme (33.4) et renvoyant une relation de la forme (33.6).
10. Expliciter (33.4) sur l'exemple $h_{n,k} = \binom{n+k}{k}^2\binom{n}{k}^2$, puis calculer les grandeurs principales intervenant dans l'algorithme : A, B, r, s, $\mathrm{H}_{n,k}$. Décrire le système linéaire à résoudre : taille, degré des coefficients.
11. De nouveau sur le cas général, par quel algorithme spécialisé du cours suggérez-vous d'effectuer l'algèbre linéaire ? En admettant que cette étape est l'étape dominante du calcul, estimer la complexité de l'algorithme, en fonction de A et B.
12. Ce qui précède ne justifie pas totalement la terminaison de l'algorithme de Zeilberger, à cause de la possible annulation du membre gauche de (33.6). Pour achever la justification, on introduit l'algèbre de Ore

$$\mathrm{S} = \mathbb{K}(n,k)\langle \partial_n, \partial_k; \mathrm{S}_n, \mathrm{S}_k, 0, 0\rangle,$$

et on remplace (33.5) par le polynôme tordu

$$\mathrm{Z} = \sum_{i=0}^{r}\sum_{j=0}^{s} f_{i,j}(n)\,\partial_n^i\partial_k^j.$$

Montrer qu'il existe un entier v, un polynôme tordu P non nul indépendant de k et ∂_k, et un polynôme tordu général Q tels que

$$\mathrm{Z} = (\partial_k - 1)^v\Big(\mathrm{P}(n,\partial_n) - (\partial_k - 1)\mathrm{Q}(n,\partial_n,\partial_k)\Big).$$

13. On introduit la factorielle descendante $k^{\underline{s}} = k(k-1)\dots(k-s+1)$, où $s \in \mathbb{N}$. Montrer la commutation, pour $a \in \mathbb{K}$,

$$(k-a)^{\underline{s}}(\partial_k - 1) = (\partial_k - 1)(k-a-1)^{\underline{s}} - s(k-s-1)^{\underline{s-1}}.$$

14. Calculer $k^{\underline{s}}(\partial_k - 1)^s$ modulo $\partial_k - 1$ à gauche.
15. En déduire que l'existence d'une relation (33.6) non triviale implique l'existence d'un $\mathrm{P} = \eta_r(n)\,\partial_n^r + \dots + \eta_0(n)$ non nul de S et d'une fraction rationnelle $\mathrm{Q} \in \mathbb{K}(n,k)$ vérifiant
$$\big(\mathrm{P} - (\partial_k - 1)\mathrm{Q}\big) \cdot h = 0.$$
16. Conclure sur l'algorithme de Zeilberger : donner une classe de termes hypergéométriques sur laquelle il termine et indiquer des bornes sur l'ordre et le degré du télescopeur obtenu.
17. L'exemple $\binom{n+k}{k}^2 \binom{n}{k}^2$ a suggéré que l'algorithme sous-jacent à ce problème n'est (tel quel) pas efficace en pratique. Suggérez ce qui permet à l'algorithme de Zeilberger d'être plus efficace.

■

Notes

L'Exercice 33.9 provient d'un article de van der Hoeven [Hoe02b] : il montre que le produit d'opérateurs différentiels de bidegrés (n,n) en (X,θ) se ramène à un nombre constant de multiplications de matrices carrées de taille n. L'article plus récent de Bostan, Chyzak et Le Roux [BCL08] montre que ces deux problèmes sont équivalents du point de vue de la complexité. L'Exercice 33.1 est inspiré par un article de Alt [Alt79], qui montre que le coût $\mathsf{R}(n)$ du calcul de la racine carrée d'une série tronquée à précision n vérifie aussi $\mathrm{O}(\mathsf{R}(n)) = \mathrm{O}(\mathsf{M}(n))$. L'Exercice 33.10 est tiré d'un article de Bostan et Salvy [BS08]. L'Exercice 33.4 provient de résultats de Kung et Tong [KT77]. L'Exercice 33.25 est une adaptation d'un article de Brent et Traub [BT80]. L'Exercice 33.14 est extrait de travaux de Bostan, Chyzak, Li et Salvy [Bos+12], l'Exercice 33.7 d'un article de Bostan, Flajolet, Salvy et Schost [Bos+06b]. La première question de l'Exercice 33.2 est tirée d'un article de Aho, Steiglitz et Ullman [ASU75], la seconde est implicite dans un papier de Bostan et Schost [BS05]. L'Exerice 33.11 est inspiré par une l'article de Keller-Gehrig [Kel85], l'Exercice 33.3 est provient d'un article de Bostan, Gaudry et Schost [BGS07].

Bibliographie

Alt79 — Alt, H. (1978/79). « Square rooting is as difficult as multiplication ». In : *Computing*, vol. 21, n°3, p. 221–232.

ASU75 — Aho, A. V., K. Steiglitz et J. D. Ullman (1975). « Evaluating polynomials at fixed sets of points ». In : *SIAM Journal on Computing*, vol. 4, n°4, p. 533–539.

BCL08 Bostan, Alin, Frédéric Chyzak et Nicolas Le Roux (2008). « Products of ordinary differential operators by evaluation and interpolation ». In : *ISSAC'08 : International Symposium on Symbolic and Algebraic Computation*. Éd. par David J. Jeffrey. ACM Press, p. 23–30.

BGS07 Bostan, Alin, Pierrick Gaudry et Éric Schost (2007). « Linear recurrences with polynomial coefficients and application to integer factorization and Cartier–Manin operator ». In : *SIAM Journal on Computing*, vol. 36, n°6, p. 1777–1806.

Bos+06b Bostan, Alin, Philippe Flajolet, Bruno Salvy et Éric Schost (2006). « Fast computation of special resultants ». In : *Journal of Symbolic Computation*, vol. 41, n°1, p. 1–29.

Bos+12 Bostan, Alin, Frédéric Chyzak, Ziming Li et Bruno Salvy (2012). « Fast computation of common left multiples of linear ordinary differential operators ». In : *ISSAC 2012 : Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 99–106.

BS05 Bostan, Alin et Éric Schost (2005). « Polynomial evaluation and interpolation on special sets of points ». In : *Journal of Complexity*, vol. 21, n°4, p. 420–446.

BS08 Borwein, Jonathan M. et Bruno Salvy (2008). « A proof of a recurrence for Bessel moments ». In : *Experimental Mathematics*, vol. 17, n°2, p. 223–230.

BT80 Brent, R. P. et J. F. Traub (1980). « On the complexity of composition and generalized composition of power series ». In : *SIAM Journal on Computing*, vol. 9, n°1, p. 54–66.

Hoe02b Hoeven, Joris van der (2002). « FFT-like multiplication of linear differential operators ». In : *Journal of Symbolic Computation*, vol. 33, n°1, p. 123–127.

Kel85 Keller-Gehrig, Walter (1985). « Fast algorithms for the characteristic polynomial ». In : *Theoretical Computer Science*, vol. 36, n°2-3, p. 309–317.

KT77 Kung, H. T. et D. M. Tong (1977). « Fast algorithms for partial fraction decomposition ». In : *SIAM Journal on Computing*, vol. 6, n°3, p. 582–593.

Bibliographie générale

Abe92 Abel, Niels Henrik (1992). *Œuvres complètes. Tome II*. Réimpression de la seconde édition (1881). Éditions Jacques Gabay. URL : http://www.gallica.fr (cité page 266).

ABP95a Abramov, Sergei A., Manuel Bronstein et Marko Petkovšek (1995). « On Polynomial Solutions of Linear Operator Equations ». In : *Symbolic and Algebraic Computation*. Éd. par A. H. M. Levelt. Proceedings of ISSAC'95, July 1995, Montreal, Canada. New York : ACM Press, p. 290–296 (cité page 310).

ABP95b — (1995). « On polynomial solutions of linear operator equations ». In : *ISSAC'95 : International Symposium on Symbolic and Algebraic Computation*. Éd. par A. H. M. Levelt. New York : ACM Press, p. 290–296 (cité page 321).

Abr03 Abramov, S. A. (2003). « When does Zeilberger's algorithm succeed ? » In : *Advances in Applied Mathematics*, vol. 30, n°3, p. 424–441 (cité page 541).

Abr89 — (1989). « Rational solutions of linear differential and difference equations with polynomial coefficients ». In : *USSR Computational Mathematics and Mathematical Physics*, vol. 29, n°11, p. 1611–1620 (cité page 310).

Abr95 — (1995). « Rational solutions of linear difference and q-difference equations with polynomial coefficients ». In : *ISSAC'95 : International Symposium on Symbolic and Algebraic Computation*. Éd. par A. H. M. Levelt. New York : ACM Press, p. 285–289 (cité page 310).

AHU74 Aho, Alfred V., John E. Hopcroft et Jeffrey D. Ullman (1974). *The design and analysis of computer algorithms*. Addison-Wesley Publishing Co. (cité pages 12, 30).

Ajt98 Ajtai, Miklós (1998). « The shortest vector problem in L2 is NP-hard for randomized reductions ». In : *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*. STOC '98. Extended abstract. New York, NY, USA : ACM, p. 10–19 (cité page 370).

AKS04 Agrawal, Manindra, Neeraj Kayal et Nitin Saxena (2004). « PRIMES is in P ». In : *Annals of Mathematics. Second Series*, vol. 160, n°2, p. 781–793 (cité page 92).

AL04 Abdeljaoued, J. et H. Lombardi (2004). *Méthodes matricielles : introduction à la complexité algébrique*. Vol. 42. Mathématiques & Applications. Springer-Verlag (cité pages 13, 180).

AL81 Alt, H. et J. van Leeuwen (1981). « The complexity of basic complex operations ». In : *Computing. Archiv für Informatik und Numerik*, vol. 27, n°3, p. 205–215 (cité page 180).

Alt79 Alt, H. (1978/79). « Square rooting is as difficult as multiplication ». In : *Computing*, vol. 21, n°3, p. 221–232 (cité page 619).

Ant79 Antoniou, A. (1979). *Digital filters : analysis and design*. McGraw-Hill Book Co. (cité page 231).

AP05 Abramov, S. A. et M. Petkovšek (2005). « Gosper's algorithm, accurate summation, and the discrete Newton–Leibniz formula ». In : *ISSAC'05 : International Symposium on Symbolic and Algebraic Computation*. ACM, New York, p. 5–12 (cité page 541).

AP07 Abramov, Sergei A. et Marko Petkovšek (2007). « Analytic solutions of linear difference equations, formal series, and bottom summation ». In : *Computer Algebra in Scientific Computing, 10th International Workshop, CASC 2007, Bonn, Germany, September 16-20, 2007, Proceedings*, p. 1–10 (cité page 541).

AP94 — (1994). « D'Alembertian solutions of linear differential and difference equations ». In : *ISSAC'94 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 169–174 (cité page 541).

Arr06 Arrondo, Enrique (2006). « Another elementary proof of the Nullstellensatz ». In : *The American Mathematical Monthly*, vol. 113, n°2, p. 169–171 (cité page 464).

AS92 Abramowitz, Milton et Irene A. Stegun, éds. (1992). *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Réimpression de l'édition de 1972. Dover Publications Inc. (cité page 266).

ASU75 Aho, A. V., K. Steiglitz et J. D. Ullman (1975). « Evaluating polynomials at fixed sets of points ». In : *SIAM Journal on Computing*, vol. 4, n°4, p. 533–539 (cité pages 108, 619).

ASZ00 Abbott, John, Victor Shoup et Paul Zimmermann (2000). « Factorization in $\mathbb{Z}[x]$: the searching phase ». In : *ISSAC'00 : International Symposium on Symbolic and Algebraic Computation*. St. Andrews, Scotland : ACM Press, p. 1–7 (cité page 396).

BA80 Bitmead, Robert R. et Brian D. O. Anderson (1980). « Asymptotically fast solution of Toeplitz and related systems of linear equations ». In : *Linear Algebra and its Applications*, vol. 34, p. 103–116 (cité page 206).

Baj+93 Bajaj, Chanderjit, John Canny, Thomas Garrity et Joe Warren (1993). « Factoring rational polynomials over the complex numbers ». In : *SIAM Journal on Computing*, vol. 22, n°2, p. 318–331 (cité page 413).

Bar99 Barkatou, Moulay A. (1999). « On rational solutions of systems of linear differential equations ». In : *Journal of Symbolic Computation*, vol. 28, n°4-5. Differential algebra and differential equations, p. 547–567 (cité page 588).

BCD16 Bostan, Alin, Gilles Christol et Philippe Dumas (2016). « Fast Computation of the Nth Term of an Algebraic Series over a Finite Prime Field ». In : *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation*. ISSAC'16. ACM, p. 119–126 (cité page 287).

BCL08 Bostan, Alin, Frédéric Chyzak et Nicolas Le Roux (2008). « Products of ordinary differential operators by evaluation and interpolation ». In : *ISSAC'08 : International Symposium on Symbolic and Algebraic Computation*. Éd. par David J. Jeffrey. ACM Press, p. 23–30 (cité page 619).

BCP13 Bostan, Alin, Frédéric Chyzak et Élie de Panafieu (2013). « Complexity estimates for two uncoupling algorithms ». In : *ISSAC 2013, Proceedings of the 38th Internatio-*

nal Symposium on Symbolic and Algebraic Computation. ACM, New York, p. 85–92 (cité page 588).

BCS05 Bostan, Alin, Thomas Cluzeau et Bruno Salvy (2005). « Fast algorithms for polynomial solutions of linear differential equations ». In : *ISSAC'05 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 45–52 (cité page 310).

BCS97 Bürgisser, Peter, Michael Clausen et M. Amin Shokrollahi (1997). *Algebraic complexity theory*. Vol. 315. Grundlehren der Mathematischen Wissenschaften. Springer-Verlag (cité pages 13, 180).

Bec+05 Beck, Matthias, Bruce C. Berndt, O-Yeat Chan et Alexandru Zaharescu (2005). « Determinations of analogues of Gauss sums and other trigonometric sums ». In : *International Journal of Number Theory*, vol. 1, n°3, p. 333–356 (cité page 30).

Bec92 Beckermann, Bernhard (1992). « A reliable method for computing M-Padé approximants on arbitrary staircases ». In : *Journal of Computational and Applied Mathematics*, vol. 40, n°1, p. 19–42 (cité page 154).

Bel+09 Belabas, Karim, Mark van Hoeij, Jürgen Klüners et Allan Steel (2009). « Factoring polynomials over global fields ». In : *Journal de Théorie des Nombres de Bordeaux*, vol. 21, n°1, p. 15–39 (cité pages 396, 413).

Bera Bernstein, Daniel J. (2001). *Multidigit multiplication for mathematicians*. url : http://cr.yp.to/papers.html (visible en 2001) (cité page 52).

Berb — (2001). *Removing redundancy in high-precision Newton iteration*. url : http://cr.yp.to/fastnewton.html (visible en 2001) (cité page 78).

Berc — (2006). *The transposition principle*. url : http://cr.yp.to/transposition.html (visible en 2006) (cité page 231).

Ber08 — (2008). « Fast multiplication and its applications ». In : *Algorithmic number theory : lattices, number fields, curves and cryptography*. Vol. 44. Publications of the Research Institute for Mathematical Sciences. Cambridge University Press, p. 325–384 (cité pages 78, 321).

Ber67 Berlekamp, E. R. (1967). « Factoring polynomials over finite fields ». In : *Bell System Technical Journal*, vol. 46, p. 1853–1859 (cité page 356).

Ber70 — (1970). « Factoring polynomials over large finite fields ». In : *Mathematics of Computation*, vol. 24, p. 713–735 (cité page 356).

Ber84 Berkowitz, Stuart J. (1984). « On computing the determinant in small parallel time using a small number of processors ». In : *Information Processing Letters*, vol. 18, p. 147–150 (cité page 181).

Ber98 Bernstein, Daniel J. (1998). « Composing power series over a finite ring in essentially linear time ». In : *Journal of Symbolic Computation*, vol. 26, n°3, p. 339–341 (cité page 77).

Béz64 Bézout, É. (1764). « Recherches sur le degré des équations résultantes de l'évanouissement des inconnues ». In : *Histoire de l'académie royale des sciences*, p. 288–338 (cité page 135).

BFS15 Bardet, Magali, Jean-Charles Faugère et Bruno Salvy (2015). « On the complexity of the F_5 Gröbner basis algorithm ». In : *Journal of Symbolic Computation*, vol. 70, p. 49–70 (cité page 481).

BG96 Baker Jr., George A. et Peter Graves-Morris (1996). *Padé approximants*. 2e éd. Vol. 59. Encyclopedia of Mathematics and its Applications. Cambridge University Press (cité page 153).

BGS07 Bostan, Alin, Pierrick Gaudry et Éric Schost (2007). « Linear recurrences with polynomial coefficients and application to integer factorization and Cartier–Manin

operator ». In : *SIAM Journal on Computing*, vol. 36, n°6, p. 1777–1806 (cité pages 286, 619).

BGS72 Beeler, Michael, R. William Gosper et Richard Schroeppel (1972). *HAKMEM*. Artificial Intelligence Memo No. 239. Massachusetts Institute of Technology (cité page 321).

BGY80 Brent, Richard P., Fred G. Gustavson et David Y. Y. Yun (1980). « Fast solution of Toeplitz systems of equations and computation of Padé approximants ». In : *Journal of Algorithms*, vol. 1, n°3, p. 259–295 (cité pages 135, 153, 206).

BH74 Bunch, James R. et John E. Hopcroft (1974). « Triangular factorization and inversion by fast matrix multiplication ». In : *Mathematics of Computation*, vol. 28, p. 231–236 (cité page 180).

BHL11 Berthomieu, Jérémy, Joris van der Hoeven et Grégoire Lecerf (2011). « Relaxed algorithms for p-adic numbers ». In : *Journal de Théorie des Nombres de Bordeaux*, vol. 23, n°3, p. 541–577 (cité page 78).

BHS80 Bentley, Jon Louis, Dorothea Haken et James B. Saxe (1980). « A general method for solving divide-and-conquer recurrences ». In : *SIGACT News*, vol. 12, n°3, p. 36–44 (cité page 30).

Bin+79 Bini, D., M. Capovani, F. Romani et G. Lotti (1979). « O($n^{2.7799}$) complexity for $n \times n$ approximate matrix multiplication ». In : *Information Processing Letters*, vol. 8, n°5, p. 234–235 (cité page 183).

Bin80 Bini, D. (1980). « Relations between exact and approximate bilinear algorithms. Applications ». In : *Calcolo*, vol. 17, n°1, p. 87–97 (cité page 183).

BJS08 Bostan, Alin, Claude-Pierre Jeannerod et Éric Schost (2008). « Solving structured linear systems with large displacement rank ». In : *Theoretical Computer Science*, vol. 407, n°1-3, p. 155–181 (cité page 206).

BK78 Brent, R. P. et H. T. Kung (1978). « Fast algorithms for manipulating formal power series ». In : *Journal of the ACM*, vol. 25, n°4, p. 581–595 (cité pages 77, 181, 247).

BKL91 Bürgisser, P., M. Karpinski et T. Lickteig (1991). « Some computational problems in linear algebra as hard as matrix multiplication ». In : *Computational Complexity*, vol. 1, n°2, p. 131–155 (cité page 181).

BL00 Beckermann, Bernhard et George Labahn (2000). « Fraction-free computation of matrix rational interpolants and matrix GCDs ». In : *SIAM Journal on Matrix Analysis and Applications*, vol. 22, n°1, p. 114–144 (cité page 154).

BL04 Bürgisser, Peter et Martin Lotz (2004). « Lower bounds on the bounded coefficient complexity of bilinear maps ». In : *Journal of the Association for Computing Machinery*, vol. 51, n°3, p. 464–482 (cité page 53).

BL94 Beckermann, B. et G. Labahn (1994). « A uniform approach for the fast computation of matrix-type Padé approximants ». In : *SIAM Journal on Matrix Analysis and Applications*, vol. 15, n°3, p. 804–823 (cité page 154).

Blä01 Bläser, Markus (2001). « A $\frac{5}{2}n^2$-lower bound for the multiplicative complexity of $n \times n$-matrix multiplication ». In : *STACS 2001 (Dresden)*. Vol. 2010. Lecture Notes in Computer Science. Springer-Verlag, p. 99–109 (cité page 182).

Blä03 — (2003). « On the complexity of the multiplication of matrices of small formats ». In : *Journal of Complexity*, vol. 19, n°1, p. 43–60 (cité page 182).

BLS03 Bostan, Alin, Grégoire Lecerf et Éric Schost (2003). « Tellegen's principle into practice ». In : *ISSAC'03 : International Symposium on Symbolic and Algebraic Computation*. Éd. par J. R. Sendra. ACM Press, p. 37–44 (cité pages 107, 230, 231).

Blu68 Bluestein, Leo I. (1968). « A linear filtering approach to the computation of the discrete Fourier transform ». In : *IEEE Northeast Electronics Research and Engineering Meeting*, vol. 10, p. 218–219 (cité page 231).

Blu70 Bluestein, L. I. (1970). « A linear filtering approach to the computation of the discrete Fourier transform ». In : *IEEE Trans. Electroacoustics*, vol. AU-18, p. 451–455 (cité page 107).

BM71 Borodin, A. et I. Munro (1971). « Evaluating polynomials at many points ». In : *Information Processing Letters*, vol. 1, n°2, p. 66–68 (cité pages 107, 181).

BM74 Borodin, A. et R. T. Moenck (1974). « Fast modular transforms ». In : *Journal of Computer and System Sciences*, vol. 8, n°3, p. 366–386 (cité page 107).

BM97 Bernardin, Laurent et Michael B. Monagan (1997). « Efficient multivariate factorization over finite fields ». In : *Applied algebra, algebraic algorithms and error-correcting codes (Toulouse, 1997)*. Vol. 1255. Lecture Notes in Computer Science. Springer-Verlag, p. 15–28 (cité page 412).

Boo72 Boole, George (1872). *A treatise on the calculus of finite differences*. 2nd. Macmillan (cité page 310).

Bor56 Bordewijk, J. L. (1956). « Inter-reciprocity applied to electrical networks ». In : *Applied Scientific Research, Section B*, vol. 6, p. 1–74 (cité page 231).

Bor85 Borwein, Peter B. (1985). « On the complexity of calculating factorials ». In : *Journal of Algorithms*, vol. 6, n°3, p. 376–380 (cité page 286).

Bos+04a Bostan, A., G. Lecerf, B. Salvy, É. Schost et B. Wiebelt (2004). « Complexity issues in bivariate polynomial factorization ». In : *ISSAC'04 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 42–49 (cité page 230).

Bos+04b Bostan, Alin, Grégoire Lecerf, Bruno Salvy, Éric Schost et Bernd Wiebelt (2004). « Complexity issues in bivariate polynomial factorization ». In : *ISSAC'04 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 42–49 (cité page 413).

Bos+06a Bostan, A., F. Chyzak, T. Cluzeau et B. Salvy (2006). « Low complexity algorithms for linear recurrences ». In : *ISSAC'06 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 31–38 (cité pages 310, 558).

Bos+06b Bostan, Alin, Philippe Flajolet, Bruno Salvy et Éric Schost (2006). « Fast computation of special resultants ». In : *Journal of Symbolic Computation*, vol. 41, n°1, p. 1–29 (cité pages 77, 619).

Bos+07a Bostan, A., F. Chyzak, F. Ollivier, B. Salvy, É. Schost et A. Sedoglavic (2007). « Fast computation of power series solutions of systems of differential equations ». In : *SODA'07 : ACM-SIAM Symposium on Discrete Algorithms*. SIAM, p. 1012–1021 (cité page 247).

Bos+07b Bostan, Alin, Frédéric Chyzak, Grégoire Lecerf, Bruno Salvy et Éric Schost (2007). « Differential equations for algebraic functions ». In : *ISSAC'07 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Dongming Wang. ACM Press, p. 25–32 (cité page 266).

Bos+12 Bostan, Alin, Frédéric Chyzak, Ziming Li et Bruno Salvy (2012). « Fast computation of common left multiples of linear ordinary differential operators ». In : *ISSAC 2012 : Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 99–106 (cité page 619).

Bos03 Bostan, Alin (2003). « Algorithmique efficace pour des opérations de base en calcul formel ». Thèse de Doctorat. École polytechnique (cité page 267).

BP02 Blondel, Vincent D. et Natacha Portier (2002). « The presence of a zero in an integer linear recurrent sequence is NP-hard to decide ». In : *Linear Algebra and its Applications*, vol. 351/352. Fourth special issue on linear systems and control, p. 91–98 (cité page 91).

BP94a Bini, Dario et Victor Y. Pan (1994). *Polynomial and matrix computations. Vol. 1*. Progress in Theoretical Computer Science. Fundamental algorithms. Birkhäuser Boston, Inc., Boston, MA, (cité page 13).

BP94b Bronstein, M. et M. Petkovšek (1994). « On Ore rings, linear operators and factorisation ». In : *Programmirovanie*, vol. 1. Also available as Research Report 200, Informatik, ETH Zürich, p. 27–44 (cité page 558).

BP96 Bronstein, Manuel et Marko Petkovšek (1996). « An introduction to pseudo-linear algebra ». In : *Theoretical Computer Science*, vol. 157, p. 3–33 (cité page 558).

Bre76 Brent, Richard P. (1976). « Multiple-precision zero-finding methods and the complexity of elementary function evaluation ». In : *Analytic computational complexity*. Proceedings of a Symposium held at Carnegie-Mellon University, Pittsburgh, PA, 1975. Academic Press, p. 151–176 (cité pages 77, 322).

Bri74 Brigham, E. Oran (1974). *The fast Fourier transform*. Prentice-Hall (cité page 52).

Bro71 Brown, W. S. (1971). « On Euclid's algorithm and the computation of polynomial greatest common divisors ». In : *SYMSAC'71 : ACM Symposium on Symbolic and Algebraic Manipulation*. Los Angeles, California, United States : ACM, p. 195–211 (cité page 135).

Bro92 Bronstein, Manuel (1992). « Linear ordinary differential equations : breaking through the order 2 barrier ». In : *ISSAC'92 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 42–48 (cité page 558).

BS04 Bostan, A. et É. Schost (2004). « On the complexities of multipoint evaluation and interpolation ». In : *Theoretical Computer Science*, vol. 329, n°1–3, p. 223–235 (cité page 231).

BS05 Bostan, Alin et Éric Schost (2005). « Polynomial evaluation and interpolation on special sets of points ». In : *Journal of Complexity*, vol. 21, n°4, p. 420–446 (cité pages 107, 108, 619).

BS08 Borwein, Jonathan M. et Bruno Salvy (2008). « A proof of a recurrence for Bessel moments ». In : *Experimental Mathematics*, vol. 17, n°2, p. 223–230 (cité page 619).

BS83 Baur, W. et V. Strassen (1983). « The complexity of partial derivatives ». In : *Theoretical Computer Science*, vol. 22, p. 317–330 (cité pages 180, 232).

BSS03 Bostan, Alin, Bruno Salvy et Éric Schost (2003). « Fast algorithms for zero-dimensional polynomial systems using duality ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 14, n°4, p. 239–272 (cité page 464).

BSS08 — (2008). « Power series composition and change of basis ». In : *ISSAC'08 : International Symposium on Symbolic and Algebraic Computation*. Éd. par David J. Jeffrey. ACM Press, p. 269–276 (cité pages 77, 78, 231).

BT71 Brown, W. S. et J. F. Traub (1971). « On Euclid's algorithm and the theory of subresultants ». In : *Journal of the Association for Computing Machinery*, vol. 18, p. 505–514 (cité page 135).

BT80 Brent, R. P. et J. F. Traub (1980). « On the complexity of composition and generalized composition of power series ». In : *SIAM Journal on Computing*, vol. 9, n°1, p. 54–66 (cité page 619).

BT86 Berkovich, L. M. et V. G. Tsirulik (1986). « Differential resultants and some of their applications ». In : *Differentsial'nye Uravneniya*, vol. 22, n°5. English translation in : Differential Equations, Plenum Publ. Corp., Vol. 22, no. 5, pp. 530–536. NY Plenum 1986, p. 750–757, 914 (cité page 558).

BT88 Ben-Or, M. et P. Tiwari (1988). « A deterministic algorithm for sparse multivariate polynomial interpolation ». In : *STOC'88 : ACM Symposium on Theory of Computing*. ACM Press, p. 301–309 (cité pages 153, 231).

Buc65 Buchberger, Bruno (1965). « Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal ». Thèse de doctorat. Mathematical Institute, University of Innsbruck, Austria (cité page 444).

BW93 Becker, Thomas et Volker Weispfenning (1993). *Gröbner bases. A computational approach to commutative algebra*. Vol. 141. Graduate Texts in Mathematics. Springer-Verlag New York (cité pages 13, 434, 444).

BZ07 Bodrato, Marco et Alberto Zanoni (2007). « Integer and polynomial multiplication : towards optimal Toom–Cook matrices ». In : *ISSAC'07 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Dongming Wang. ACM Press, p. 17–24 (cité page 53).

CA69 Cook, S. A. et S. O. Aanderaa (1969). « On the minimum computation time of functions ». In : *Transactions of the American Mathematical Society*, vol. 142, p. 291–314 (cité page 53).

Cam83 Camion, Paul (1983). « A deterministic algorithm for factorizing polynomials of $\mathbf{F}_q[X]$ ». In : *Combinatorial mathematics (Marseille-Luminy, 1981)*. Vol. 75. North-Holland Math. Stud. North-Holland, Amsterdam, p. 149–157 (cité page 356).

Car92 Cartier, Pierre (1992). « Démonstration "automatique" d'identités et fonctions hypergéométriques (d'après D. Zeilberger) ». In : *Astérisque*, vol. 1991/92, n°206. Séminaire Bourbaki, Exp. No. 746, 3, 41–91 (cité page 558).

Car99 Cardinal, Jean-Paul (1999). « On a property of Cauchy-like matrices ». In : *Comptes Rendus de l'Académie des Sciences. Série I. Mathématique*, vol. 328, n°11, p. 1089–1093 (cité page 206).

CC86 Cabay, Stanley et Dong Koo Choi (1986). « Algebraic computations of scaled Padé fractions ». In : *SIAM Journal on Computing*, vol. 15, n°1, p. 243–270 (cité page 153).

CC88 Chudnovsky, D. V. et G. V. Chudnovsky (1988). « Approximations and complex multiplication according to Ramanujan ». In : *Proceedings of the Centenary Conference, University of Illinois at Urbana-Champaign, June 1–5, 1987*. Éd. par G. E. Andrews, R. A. Askey, B. C. Berndt, K. G. Ramanathan et R. A. Rankin. Academic Press, p. 375–472 (cité pages 286, 322).

CG00 Chu, Eleanor et Alan George (2000). *Inside the FFT black box*. Serial and parallel fast Fourier transform algorithms. CRC Press (cité page 231).

CH07 Chung, Jaewook et M. Anwar Hasan (2007). « Asymmetric squaring formulae ». In : *ARITH'07 : IEEE Symposium on Computer Arithmetic*. IEEE, p. 113–122 (cité page 53).

CH14 Costa, Edgar et David Harvey (2014). « Faster deterministic integer factorization ». In : *Mathematics of Computation*, vol. 83, n°285, p. 339–345 (cité page 286).

Cha91 Chardin, Marc (1991). « Differential resultants and subresultants ». In : *Fundamentals of computation theory*. Vol. 529. Lecture Notes in Computer Science. Gosen : Springer-Verlag, p. 180–189 (cité page 558).

Chè04 Chèze, Guillaume (2004). « Des méthodes symboliques-numériques et exactes pour la factorisation absolue des polynômes en deux variables ». Thèse de doctorat. Université de Nice-Sophia Antipolis (France) (cité page 413).

Che84 Cheng, Unjeng (1984). « On the continued fraction and Berlekamp's algorithm ». In : *IEEE Transactions on Information Theory*, vol. 30, n°3, p. 541–544 (cité page 153).

Chy00 Chyzak, Frédéric (2000). « An extension of Zeilberger's fast algorithm to general holonomic functions ». In : *Discrete Mathematics*, vol. 217, n°1-3, p. 115–134 (cité page 588).

CK91 Cantor, D. G. et E. Kaltofen (1991). « On fast multiplication of polynomials over arbitrary algebras ». In : *Acta Informatica*, vol. 28, n°7, p. 693–701 (cité page 53).

CKM97 Collart, S., M. Kalkbrener et D. Mall (1997). « Converting bases with the Gröbner walk ». In : *Journal of Symbolic Computation*, vol. 24, n°3-4, p. 465–469 (cité page 444).

CKO09 Cenk, M., C. K. Koc et F. Ozbudak (2009). « Polynomial multiplication over finite fields using field extensions and interpolation ». In : *ARITH'09 : IEEE Symposium on Computer Arithmetic*. IEEE Computer Society, p. 84–91 (cité page 54).

CKY89 Canny, John F., Erich Kaltofen et Lakshman Yagati (1989). « Solving systems of non-linear polynomial equations faster ». In : *ISSAC'89 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 121–128 (cité pages 231, 232).

CL89 Cabay, S. et G. Labahn (1989). « A fast, reliable algorithm for calculating Padé–Hermite forms ». In : *ISSAC'89 : International Symposium on Symbolic and Algebraic Computation*. Portland, Oregon, United States : ACM Press, p. 95–100 (cité page 154).

CLB92 Cabay, S., G. Labahn et B. Beckermann (1992). « On the theory and computation of nonperfect Padé–Hermite approximants ». In : *Journal of Computational and Applied Mathematics*, vol. 39, n°3, p. 295–313 (cité page 154).

CLO96 Cox, David, John Little et Donal O'Shea (1996). *Ideals, varieties, and algorithms. An introduction to computational algebraic geometry and commutative algebra*. 2e éd. Undergraduate Texts in Mathematics. Springer-Verlag (cité pages 13, 434, 444, 480).

CMP87 Cerlienco, L., M. Mignotte et F. Piras (1987). « Suites récurrentes linéaires. Propriétés algébriques et arithmétiques ». In : *L'Enseignement Mathématique*. II, vol. 33, p. 67–108 (cité pages 91, 230).

Coh+05 Cohn, H., R. Kleinberg, B. Szegedy et C. Umans (2005). « Group-theoretic algorithms for matrix multiplication ». In : *FOCS'05 : IEEE Conference on Foundations of Computer Science*. IEEE Computer Society, p. 379–388 (cité page 184).

Col66 Collins, G. E. (1966). « Polynomial remainder sequences and determinants ». In : *The American Mathematical Monthly*, vol. 73, p. 708–712 (cité page 135).

Col67 Collins, George E. (1967). « Subresultants and reduced polynomial remainder sequences ». In : *Journal of the Association for Computing Machinery*, vol. 14, p. 128–142 (cité page 135).

Col71 — (1971). « The calculation of multivariate polynomial resultants ». In : *Journal of the Association for Computing Machinery*, vol. 18, p. 515–532 (cité page 135).

Com70 Comtet, L. (1970). *Analyse combinatoire*. 2 volumes. Presses Universitaires de France (cité page 266).

Coo66 Cook, S. (1966). « On the minimum computation time of functions ». Thèse de doctorat. Harvard University (cité page 52).

Coo90 Cooley, James W. (1990). « How the FFT gained acceptance ». In : *A history of scientific computing*. ACM Press Hist. Ser. P : ACM, p. 133–140 (cité page 52).

Cop94 Coppersmith, Don (1994). « Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm ». In : *Mathematics of Computation*, vol. 62, n°205, p. 333–350 (cité page 193).

Cor+09 Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest et Clifford Stein (2009). *Introduction to algorithms*. Third. MIT Press, Cambridge, MA (cité pages 12, 30).

CPS08 Chen, William Y. C., Peter Paule et Husam L. Saad (2008). « Converging to Gosper's algorithm ». In : *Advances in Applied Mathematics*, vol. 41, n°3, p. 351–364 (cité page 310).

CS98 Chyzak, Frédéric et Bruno Salvy (1998). « Non-commutative elimination in Ore algebras proves multivariate holonomic identities ». In : *Journal of Symbolic Computation*, vol. 26, n°2, p. 187–227 (cité pages 558, 576).

CT65 Cooley, James W. et John W. Tukey (1965). « An algorithm for the machine calculation of complex Fourier series ». In : *Mathematics of Computation*, vol. 19, p. 297–301 (cité page 52).

CT93 — (1993). « On the origin and publication of the FFT paper ». In : *Current Contents*, vol. 33, n°51–52, p. 8–9 (cité page 52).

CU03 Cohn, Henry et Christopher Umans (2003). « A group-theoretic approach to fast matrix multiplication ». In : *FOCS'03 : IEEE Conference on Foundations of Computer Science*. Washington, DC, USA : IEEE Computer Society, p. 438 (cité page 184).

CW90 Coppersmith, Don et Shmuel Winograd (1990). « Matrix multiplication via arithmetic progressions ». In : *Journal of Symbolic Computation*, vol. 9, n°3, p. 251–280 (cité page 183).

CZ81 Cantor, David G. et Hans Zassenhaus (1981). « A new algorithm for factoring polynomials over finite fields ». In : *Mathematics of Computation*, vol. 36, n°154, p. 587–592 (cité page 356).

Dan37 Danilevskii, A. M (1937). « The numerical solution of the secular equation ». In : *Matematicheskiĭ Sbornik*, vol. 44, n°2. (in Russian), p. 169–171 (cité page 181).

DD84a Della Dora, J. et C. Dicrescenzo (1984). « Approximants de Padé–Hermite. I. Théorie ». In : *Numerische Mathematik*, vol. 43, n°1, p. 23–39 (cité page 154).

DD84b — (1984). « Approximants de Padé–Hermite. II. Programmation ». In : *Numerische Mathematik*, vol. 43, n°1, p. 41–57 (cité page 154).

De+08 De, A., P. P. Kurur, C. Saha et R. Saptharishi (2008). « Fast integer multiplication using modular arithmetic ». In : *STOC'08 : ACM Symposium on Theory of Computing*. Victoria, British Columbia, Canada : ACM, p. 499–506 (cité page 53).

Dem87 Demazure, M. (1987). « Le théorème de complexité de Mayr et Meyer ». In : *Géométrie algébrique et applications, I (La Rábida, 1984)*. Vol. 22. Travaux en Cours. Paris : Hermann, p. 35–58 (cité page 481).

Dem97 Demazure, Michel (1997). *Cours d'algèbre*. Nouvelle Bibliothèque Mathématique n°1. Cassini, Paris (cité page 53).

Der94 Derksen, Harm (1994). *An algorithm to compute generalized Padé–Hermite forms*. Report n°9403. Dept. of Math., Catholic University Nijmegen (cité page 153).

Dix82 Dixon, John D. (1982). « Exact solution of linear equations using p-adic expansions ». In : *Numerische Mathematik*, vol. 40, n°1, p. 137–141 (cité pages 153, 214).

DL07 Durvye, Clémence et Grégoire Lecerf (2007). « A concise proof of the Kronecker polynomial system solver from scratch ». In : *Expositiones Mathematicae*, vol. 26, n°2, p. 101–139 (cité pages 500, 522, 523).

DL78 DeMillo, Richard A. et Richard J. Lipton (1978). « A probabilistic remark on algebraic program testing ». In : *Information Processing Letters*, vol. 7, n°4, p. 193–195 (cité page 193).

Dor87 Dornstetter, Jean-Louis (1987). « On the equivalence between Berlekamp's and Euclid's algorithms ». In : *IEEE Transactions on Information Theory*, vol. 33, n°3, p. 428–431 (cité page 153).

DPR61 Davis, Martin, Hilary Putnam et Julia Robinson (1961). « The decision problem for exponential diophantine equations ». In : *Annals of Mathematics. Second Series*, vol. 74, p. 425–436 (cité page 30).

DS00 Dongarra, J. et F. Sullivan (2000). « Top ten algorithms ». In : *Computing in Science & Engineering*, vol. 2, n°1, p. 22–23 (cité page 52).

DS11 Drmota, Michael et Wojciech Szpankowski (2011). « A master theorem for discrete divide and conquer recurrences ». In : *SODA'11 : ACM-SIAM Symposium on Discrete Algorithms*. San Francisco, CA : SIAM, p. 342–361 (cité page 30).

Dur08 Durvye, Clémence (2008). « Algorithmes pour la décomposition primaire des idéaux polynomiaux de dimension nulle donnés en évaluation ». Thèse de doctorat. Université de Versailles Saint-Quentin-en-Yvelines (cité page 522).

DV90 Duhamel, P. et M. Vetterli (1990). « Fast Fourier transforms : a tutorial review and a state of the art ». In : *Signal Processing. An Interdisciplinary Journal*, vol. 19, n°4, p. 259–299 (cité page 52).

Eis95 Eisenbud, David (1995). *Commutative algebra. With a view toward algebraic geometry*. Vol. 150. Graduate Texts in Mathematics. Springer-Verlag New York (cité pages 13, 434).

EKM96 Erlingsson, Úlfar, Erich Kaltofen et David Musser (1996). « Generic Gram–Schmidt orthogonalization by exact division ». In : *Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation*. ISSAC'96. New York, NY, USA : ACM, p. 275–282 (cité page 370).

El 03 El Kahoui, M'hammed (2003). « An elementary approach to subresultants theory ». In : *Journal of Symbolic Computation*, vol. 35, n°3, p. 281–292 (cité page 135).

Erd32 Erdős, Paul (1932). « Beweis eines Satzes von Tschebyschef ». In : *Acta Litterarum ac Scientiarum, Szeged*, n°5, p. 194–198 (cité page 396).

Evd94 Evdokimov, Sergei (1994). « Factorization of polynomials over finite fields in subexponential time under GRH ». In : *Algorithmic number theory (Ithaca, NY, 1994)*. Vol. 877. Lecture Notes in Computer Science. Springer-Verlag, p. 209–219 (cité page 356).

Eve+03 Everest, Graham, Alf van der Poorten, Igor Shparlinski et Thomas Ward (2003). *Recurrence sequences*. Vol. 104. Mathematical Surveys and Monographs. American Mathematical Society (cité page 91).

Fau+93 Faugère, J. C., P. Gianni, D. Lazard et T. Mora (1993). « Efficient computation of zero-dimensional Gröbner bases by change of ordering ». In : *Journal of Symbolic Computation*, vol. 16, n°4, p. 329–344 (cité pages 444, 576).

Fau02 Faugère, Jean-Charles (2002). « A new efficient algorithm for computing Gröbner bases without reduction to zero (F5) ». In : *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*. ISSAC'02. New York, NY, USA : ACM, p. 75–83 (cité pages 445, 481).

Fau99 — (1999). « A new efficient algorithm for computing Gröbner bases (F4) ». In : *Journal of Pure and Applied Algebra*, vol. 139, n°1–3, p. 61–88 (cité page 445).

Fet71 Fettweis, A. (1971). « A general theorem for signal-flow networks, with applications ». In : *Archiv für Elektronik und Übertragungstechnik*, vol. 25, n°12, p. 557–561 (cité page 231).

FGP01 Flajolet, P., X. Gourdon et D. Panario (2001). « The complete analysis of a polynomial factorization algorithm over finite fields ». In : *Journal of Algorithms*, vol. 40, n°1, p. 37–81 (cité page 356).

FGS95 Fitchas, N., M. Giusti et F. Smietanski (1995). « Sur la complexité du théorème des zéros ». In : *Approximation and optimization in the Caribbean, II (Havana, 1993)*. Vol. 8. Approximation & optimization. Frankfurt am Main : P. Lang, p. 274–329 (cité page 522).

Fid72a Fiduccia, C. M. (1972). « On obtaining upper bounds on the complexity of matrix multiplication ». In : *Complexity of computer computations*. IBM Thomas J. Watson Research Center, Yorktown Heights, New York : Plenum, p. 31–40 (cité page 231).

Fid72b — (1972). « Polynomial evaluation via the division algorithm : the fast Fourier transform revisited. » In : *STOC'72 : ACM Symposium on Theory of Computing*, p. 88–93 (cité page 107).

Fid72c Fiduccia, Charles M. (1972). « On obtaining upper bounds on the complexity of matrix multiplication ». In : *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*. New York : Plenum, p. 31–40 (cité page 180).

Fid73 Fiduccia, C. M. (1973). « On the algebraic complexity of matrix multiplication ». Thèse de doctorat. Brown Univ., Providence, RI, Center Comput. Inform. Sci., Div. Engin. (cité page 231).

Fid85 — (1985). « An efficient formula for linear recurrences ». In : *SIAM Journal on Computing*, vol. 14, n°1, p. 106–112 (cité page 92).

Fis74 Fischer, Patrick C. (1974). « Further schemes for combining matrix algorithms ». In : *Automata, languages and programming*. Vol. 14. Lecture Notes in Computer Science. Springer-Verlag, p. 428–436 (cité page 182).

FJ05 Frigo, M. et S. G. Johnson (2005). « The design and implementation of FFTW3 ». In : *Proceedings of the IEEE*, vol. 93, n°2, p. 216–231 (cité pages 52, 53).

FP74 Fischer, P. C. et R. L. Probert (1974). « Efficient procedures for using matrix algorithms ». In : *Automata, languages and programming*. Vol. 14. Lecture Notes in Computer Science. Springer-Verlag, p. 413–427 (cité page 182).

FS56 Fröhlich, A. et J. C. Shepherdson (1956). « Effective procedures in field theory ». In : *Philosophical Transactions of the Royal Society A*, vol. 248, p. 407–432 (cité page 342).

FS82 Flajolet, Philippe et Jean-Marc Steyaert (1982). « A branching process arising in dynamic hashing, trie searching and polynomial factorization ». In : *Automata, Languages and Programming*. Éd. par M. Nielsen et E. Schmidt. Vol. 140. Lecture Notes in Computer Science. Springer-Verlag, p. 239–251 (cité page 356).

FS97 Flajolet, Philippe et Bruno Salvy (1997). « The SIGSAM challenges : symbolic asymptotics in practice ». In : *ACM SIGSAM Bulletin*, vol. 31, n°4, p. 36–47 (cité page 286).

Für07 Fürer, Martin (2007). « Faster integer multiplication ». In : *STOC'07 : ACM Symposium on Theory of Computing*. New York : ACM, p. 57–66 (cité page 53).

Für09 — (2009). « Faster integer multiplication ». In : *SIAM Journal on Computing*, vol. 39, n°3, p. 979–1005 (cité page 53).

Gao03 Gao, Shuhong (2003). « Factoring multivariate polynomials via partial differential equations ». In : *Mathematics of Computation*, vol. 72, n°242, p. 801–822 (cité pages 412, 413).

Gar59 Garner, Harvey L. (1959). « The residue number system ». In : *IRE-AIEE-ACM'59 Western joint computer conference*. San Francisco, California : ACM, p. 146–153 (cité page 107).

Gat06 Gathen, Joachim von zur (2006). « Who was who in polynomial factorization ». In : *ISSAC'06 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 1–2 (cité pages 342, 412).

Gat84 — (1984). « Hensel and Newton methods in valuation rings ». In : *Mathematics of Computation*, vol. 42, n°166, p. 637–661 (cité pages 342, 412).

Gat85 Gathen, J. von zur (1985). « Irreducibility of multivariate polynomials ». In : *Journal of Computer and System Sciences*, vol. 31, n°2. Special issue : Twenty-fourth annual symposium on the foundations of computer science (Tucson, Ariz., 1983), p. 225–264 (cité page 413).

Gat87 Gathen, Joachim von zur (1987). « Factoring polynomials and primitive elements for special primes ». In : *Theoretical Computer Science*, vol. 52, n°1-2, p. 77–89 (cité page 356).

Gau66 Gauss, Carl Friedrich (1966). *Disquisitiones arithmeticae*. Translated into English by Arthur A. Clarke, S. J. Yale University Press (cité page 107).

GCL92 Geddes, Keith O., Stephen R. Czapor et George Labahn (1992). *Algorithms for computer algebra*. Kluwer Academic Publishers (cité pages 13, 135).

Ger01 Gerhard, Jürgen (2001). « Fast Modular Algorithms for Squarefree Factorization and Hermite Integration ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 11, n°3, p. 203–226 (cité page 396).

GG03 Gathen, Joachim von zur et Jürgen Gerhard (2003). *Modern computer algebra*. 2e éd. Cambridge University Press (cité pages 13, 53, 135, 342, 356, 370, 413, 481).

GG13 — (2013). *Modern computer algebra*. 3e éd. Cambridge University Press (cité page 136).

GG99 — (1999). *Modern computer algebra*. Cambridge University Press (cité page 231).

GH93 Giusti, Marc et Joos Heintz (1993). « La détermination des points isolés et de la dimension d'une variété algébrique peut se faire en temps polynomial ». In : *Computational algebraic geometry and commutative algebra (Cortona, 1991)*. Vol. XXXIV. Sympos. Math. Cambridge : Cambridge Univ. Press, p. 216–256 (cité pages 500, 522).

GHL15a Grenet, Bruno, Joris van der Hoeven et Grégoire Lecerf (2015). « Deterministic root finding over finite fields using Graeffe transforms ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 27, n°3, p. 237–257 (cité page 356).

GHL15b — (2015). « Randomized root finding over finite FFT-fields using tangent Graeffe transforms ». In : *ISSAC'15 : International Symposium on Symbolic and Algebraic Computation*. Éd. par D. Robertz. ACM Press, p. 197–204 (cité page 356).

GHS93 Giusti, Marc, Joos Heintz et Juan Sabia (1993). « On the efficiency of effective Nullstellensätze ». In : *Computational Complexity*, vol. 3, n°1, p. 56–95 (cité page 522).

Gie01 Giesbrecht, Mark (2001). « Fast computation of the Smith form of a sparse integer matrix ». In : *Computational Complexity*, vol. 10, n°1, p. 41–69 (cité page 193).

Gie95 — (1995). « Nearly optimal algorithms for canonical matrix forms ». In : *SIAM Journal on Computing*, vol. 24, n°5, p. 948–969 (cité page 181).

Gio+91 Giovini, Alessandro, Teo Mora, Gianfranco Niesi, Lorenzo Robbiano et Carlo Traverso (1991). « "One sugar cube, please" or selection strategies in the Buchberger algorithm ». In : *Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation*. ISSAC'91. New York, NY, USA : ACM, p. 49–54 (cité page 444).

Giu+00 Giusti, Marc, Klemens Hägele, Grégoire Lecerf, Joël Marchand et Bruno Salvy (2000). « The Projective Noether Maple package : computing the dimension of a projective variety ». In : *Journal of Symbolic Computation*, vol. 30, n°3, p. 291–307 (cité page 500).

Giu+95 Giusti, M., J. Heintz, J. E. Morais et L. M. Pardo (1995). « When polynomial equation systems can be "solved" fast? » In : *Applied algebra, algebraic algorithms and error-correcting codes (Paris, 1995)*. Vol. 948. Lecture Notes in Computer Science. Springer-Verlag, p. 205–231 (cité page 522).

Giu+97 Giusti, M., K. Hägele, J. Heintz, J. L. Montaña, J. E. Morais et L. M. Pardo (1997). « Lower bounds for Diophantine approximations ». In : *Journal of Pure and Applied Algebra*, vol. 117/118, p. 277–317 (cité page 522).

Giu+98 Giusti, M., J. Heintz, J. E. Morais, J. Morgenstern et L. M. Pardo (1998). « Straight-line programs in geometric elimination theory ». In : *Journal of Pure and Applied Algebra*, vol. 124, n°1-3, p. 101–146 (cité page 522).

Giu84 Giusti, M. (1984). « Some effectivity problems in polynomial ideal theory ». In : *EUROSAM 84 : International Symposium on Symbolic and Algebraic Computation Cambridge, England, July 9–11, 1984*. Éd. par John Fitch. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 159–171 (cité page 481).

Giu88 Giusti, Marc (1988). « Combinatorial dimension theory of algebraic varieties ». In : *Journal of Symbolic Computation*, vol. 6, n°2-3. Special issue on Computational aspects of commutative algebra, p. 249–265 (cité page 500).

GJV03 Giorgi, Pascal, Claude-Pierre Jeannerod et Gilles Villard (2003). « On the complexity of polynomial matrix computations ». In : *ISSAC'03 : International Symposium on Symbolic and Algebraic Computation*. Éd. par J. R. Sendra. New York : ACM Press, p. 135–142 (cité pages 154, 214).

GK85a Gathen, J. von zur et E. Kaltofen (1985). « Factorization of multivariate polynomials over finite fields ». In : *Mathematics of Computation*, vol. 45, n°171, p. 251–261 (cité page 412).

GK85b Gathen, Joachim von zur et Erich Kaltofen (1985). « Factoring sparse multivariate polynomials ». In : *Journal of Computer and System Sciences*, vol. 31, n°2. Special issue : Twenty-fourth annual symposium on the foundations of computer science (Tucson, Ariz., 1983), p. 265–287 (cité page 413).

GL02 Gao, Shuhong et Alan G. B. Lauder (2002). « Hensel lifting and bivariate polynomial factorisation over finite fields ». In : *Mathematics of Computation*, vol. 71, n°240, p. 1663–1676 (cité page 412).

GL03 Gathen, Joachim von zur et Thomas Lücking (2003). « Subresultants revisited ». In : *Theoretical Computer Science*, vol. 297, n°1-3, p. 199–239 (cité page 135).

GLM91 Gilbert, J.-C., G. Le Vey et J. Masse (1991). *La différentiation automatique de fonctions représentées par des programmes*. Rapp. tech. RR INRIA 1557 (cité page 231).

GLS01 Giusti, Marc, Grégoire Lecerf et Bruno Salvy (2001). « A Gröbner free alternative for polynomial system solving ». In : *Journal of Complexity*, vol. 17, n°1, p. 154–211 (cité page 522).

GLS98 Giesbrecht, M., A. Lobo et B. D. Saunders (1998). « Certifying inconsistency of sparse linear systems ». In : *ISSAC'98 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 113–119 (cité page 193).

GM89 Gianni, Patrizia et Teo Mora (1989). « Algebraic solution of systems of polynomial equations using Groebner bases ». In : *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes. 5th International Conference, AAECC-5, Menorca, Spain, June 15-19, 1987. Proceedings*. Éd. par Llorenc Huguet et Alain Poli. Vol. 356. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, p. 247–257 (cité page 464).

GO94 Gohberg, I. et V. Olshevsky (1994). « Complexity of multiplication with vectors for structured matrices ». In : *Linear Algebra and its Applications*, vol. 202, p. 163–192 (cité page 206).

Gos78 Gosper, R. William (1978). « Decision procedure for indefinite hypergeometric summation ». In : *Proceedings of the National Academy of Sciences USA*, vol. 75, n°1, p. 40–42 (cité page 541).

Gra+82 Gragg, William B., Fred G. Gustavson, Daniel D. Warner et David Y. Y. Yun (1982). « On fast computation of superdiagonal Padé fractions ». In : *Mathematical Programming Study*, n°18. Algorithms and theory in filtering and control (Lexington, Ky., 1980), p. 39–42 (cité page 153).

Gri90 Grigoriev, D. Yu. (1990). « Complexity of factoring and calculating the GCD of linear ordinary differential operators ». In : *Journal of Symbolic Computation*, vol. 10, n°1, p. 7–37 (cité page 558).

GS66 Gentleman, W. M. et G. Sande (1966). « Fast Fourier transforms : for fun and profit ». In : *AFIPS'66*. San Francisco, California : ACM, p. 563–578 (cité page 52).

GS72 Gohberg, I. C. et A. A. Semencul (1972). « On the inversion of finite Toeplitz matrices and their continuous analogues (in Russian) ». In : *Matematicheskie Issledovaniya*, vol. 7, n°2, p. 201–223 (cité page 206).

GT96 Gianni, P. et B. Trager (1996). « Square-free algorithms in positive characteristic ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 7, n°1, p. 1–14 (cité page 342).

Gup+12 Gupta, Somit, Soumojit Sarkar, Arne Storjohann et Johnny Valeriote (2012). « Triangular x-basis decompositions and derandomization of linear algebra algorithms over K[x] ». In : *Journal of Symbolic Computation*, vol. 47, n°4, p. 422–453 (cité page 214).

GV13 Golub, Gene H. et Charles F. Van Loan (2013). *Matrix computations*. Fourth. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD (cité page 206).

GY79 Gustavson, Fred G. et David Y. Y. Yun (1979). « Fast algorithms for rational Hermite approximation and solution of Toeplitz systems ». In : *IEEE Transactions on Circuits and Systems*, vol. 26, n°9, p. 750–755 (cité pages 135, 153, 206).

Har10 Harvey, David (2010). « Faster exponentials of power series ». url : https://arxiv.org/abs/0911.3110 (cité page 78).

Har11 — (2011). « Faster algorithms for the square root and reciprocal of power series ». In : *Mathematics of Computation*, vol. 80, p. 387–394 (cité page 78).

Har62 Harley, Rev. Robert (1862). « On the theory of the transcendental solution of algebraic equations ». In : *Quarterly Journal of Pure and Applied Mathematics*, vol. 5, p. 337–360 (cité page 266).

Har72 Harter, Richard (1972). « The optimality of Winograd's formula ». In : *Communications of the ACM*, vol. 15, n°5, p. 352 (cité page 181).

Hei+00 Heintz, Joos, Teresa Krick, Susana Puddu, Juan Sabia et Ariel Waissbein (2000). « Deformation techniques for efficient polynomial equation solving ». In : *Journal of Complexity*, vol. 16, n°1, p. 70–109 (cité page 522).

Her73a Hermite, C. (1873). « Extrait d'une lettre de Monsieur Ch. Hermite à Monsieur Paul Gordan ». In : *Journal für die Reine und Angewandte Mathematik*, vol. 78, p. 303–311 (cité page 154).

Her73b — (1873). « Sur la fonction exponentielle ». In : *Comptes-rendus de l'Académie des sciences de Paris*, vol. 77, p. 18–24 (cité page 154).

Her93 — (1893). « Sur la généralisation des fractions continues algébriques ». In : *Annali di Matematica pura ed applicata*, vol. 21, n°2, p. 289–308 (cité page 154).

HH71 Heindel, L. E. et E. Horowitz (1971). « On decreasing the computing time for modular arithmetic ». In : *12th symposium on switching and automata theory*. IEEE, p. 126–128 (cité page 108).

HHL14 Harvey, David, Joris van der Hoeven et Grégoire Lecerf (2014). « Faster polynomial multiplication over finite fields ». http://arxiv.org/abs/1407.3361 (cité page 53).

HHL16 — (2016). « Even faster integer multiplication ». In : *Journal of Complexity*, vol. 36, p. 1–30 (cité page 53).

HHN11 Hart, William, Mark van Hoeij et Andrew Novocin (2011). « Practical polynomial factoring in polynomial time ». In : *ISSAC'11 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 163–170 (cité page 397).

Hil92 Hilbert, David (1892). « Über die Irreducibilität ganzer rationaler Functionen mit ganzzahligen Coefficienten ». In : *Journal für die reine und angewandte Mathematik*, vol. 110 (cité page 413).

Hir64 Hironaka, Heisuke (1964). « Resolution of singularities of an algebraic variety over a field of characteristic zero. I, II ». In : *Annals of Mathematics. Second Series*, vol. 79, p. 205–326 (cité page 434).

HJB85 Heideman, Michael T., Don H. Johnson et C. Sidney Burrus (1985). « Gauss and the history of the fast Fourier transform ». In : *Archive for History of Exact Sciences*, vol. 34, n°3, p. 265–277 (cité page 52).

HK71 Hopcroft, J. E. et L. R. Kerr (1971). « On minimizing the number of multiplications necessary for matrix multiplication ». In : *SIAM Journal on Applied Mathematics*, vol. 20, p. 30–36 (cité page 182).

HM73 Hopcroft, J. et J. Musinski (1973). « Duality applied to the complexity of matrix multiplication and other bilinear forms ». In : *SIAM Journal on Computing*, vol. 2, p. 159–173 (cité pages 183, 231, 232).

HMW01 Heintz, Joos, Guillermo Matera et Ariel Waissbein (2001). « On the time–space complexity of geometric elimination procedures ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 11, n°4, p. 239–296 (cité page 522).

Hoe02a Hoeij, Mark van (2002). « Factoring polynomials and the knapsack problem ». In : *Journal of Number Theory*, vol. 95, n°2, p. 167–189 (cité page 396).

Hoe02b Hoeven, Joris van der (2002). « FFT-like multiplication of linear differential operators ». In : *Journal of Symbolic Computation*, vol. 33, n°1, p. 123–127 (cité page 619).

Hoe02c — (2002). « Relax, but don't be too lazy ». In : *Journal of Symbolic Computation*, vol. 34, n°6, p. 479–542 (cité pages 77, 78, 247).

Hoe03 — (2003). « Relaxed multiplication using the middle product ». In : *ISSAC'03 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Manuel Bronstein. Philadelphia, USA : ACM Press, p. 143–147 (cité page 78).

Hoe04 — (2004). « The truncated Fourier transform and applications ». In : *ISSAC'04 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 290–296 (cité page 54).

Hoe07 — (2007). « New algorithms for relaxed multiplication ». In : *Journal of Symbolic Computation*, vol. 42, n°8, p. 792–802 (cité page 78).

Hoe10 — (2010). « Newton's method and FFT trading ». In : *Journal of Symbolic Computation*, vol. 45, n°8, p. 857–878 (cité pages 78, 247).

Hoe98 Hoeij, Mark van (1998). « Rational solutions of linear difference equations ». In : *ISSAC'98 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 120–123 (cité page 310).

Hon01a Hong, Hoon (2001). « Ore principal subresultant coefficients in solutions ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 11, n°3, p. 227–237 (cité page 558).

Hon01b — (2001). « Ore subresultant coefficients in solutions ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 12, n°5, p. 421–428 (cité page 558).

Hor72 Horowitz, E. (1972). « A fast method for interpolation using preconditioning ». In : *Information Processing Letters*, vol. 1, n°4, p. 157–163 (cité page 107).

HQZ04 Hanrot, Guillaume, Michel Quercia et Paul Zimmermann (2004). « The middle product algorithm I ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 14, n°6, p. 415–438 (cité pages 54, 231, 232, 247).

HS81 Heintz, Joos et Malte Sieveking (1981). « Absolute primality of polynomials is decidable in random polynomial time in the number of variables ». In : *Automata, languages and programming (Akko, 1981)*. Vol. 115. Lecture Notes in Computer Science. Springer-Verlag, p. 16–28 (cité page 413).

HS82 Heintz, J. et C.-P. Schnorr (1982). « Testing polynomials which are easy to compute ». In : *Logic and algorithmic (Zurich, 1980)*. Vol. 30. Monograph. Enseign. Math. Geneva : Univ. Genève, p. 237–254 (cité page 523).

HS85 Harris, William A. et Yasutaka Sibuya (1985). « The reciprocals of solutions of linear ordinary differential equations ». In : *Advances in Mathematics*, vol. 58, n°2, p. 119–132 (cité page 267).

Hua91 Huang, Ming-Deh A. (1991). « Generalized Riemann hypothesis and factoring polynomials over finite fields ». In : *Journal of Algorithms*, vol. 12, n°3, p. 464–481 (cité page 356).

Hus+96 Huss-Lederman, S., E. M. Jacobson, A. Tsao, T. Turnbull et J. R. Johnson (1996). « Implementation of Strassen's algorithm for matrix multiplication ». In : *Supercomputing'96*. 32 pp. Pittsburgh, PA : IEEE Computer Society (cité page 182).

Huỳ86 Huỳnh, Dũng T. (1986). « A superexponential lower bound for Gröbner bases and Church-Rosser commutative Thue systems ». In : *Information and Control*, vol. 68, n°1-3, p. 196–206 (cité page 481).

HW08 Hardy, G. H. et E. M. Wright (2008). *An introduction to the theory of numbers*. 6e éd. Oxford University Press (cité page 396).

HZ Hanrot, Guillaume et Paul Zimmermann (2004). *Newton iteration revisited*. url : `http://www.loria.fr/~zimmerma/papers` (visible en 2004) (cité page 78).

Inc56 Ince, E. L. (1956). *Ordinary differential equations*. Reprint of the 1926 edition. Dover Publications (cité page 321).

JaJ79 Ja'Ja', Joseph (1979). « On the complexity of bilinear forms with commutativity ». In : *STOC'79 : ACM Symposium on Theory of Computing*. Atlanta, Georgia, United States : ACM, p. 197–208 (cité page 181).

Jan20 Janet, Maurice (1920). « Sur les systèmes d'équations aux dérivées partielles ». In : *Journal de Mathématiques Pures et Appliquées*, vol. 3, p. 65–151 (cité page 444).

Jer+04 Jeronimo, Gabriela, Teresa Krick, Juan Sabia et Martín Sombra (2004). « The computational complexity of the Chow form ». In : *Foundations of Computational Mathematics*, vol. 4, n°1, p. 41–117 (cité page 523).

Joh15 Johansson, Fredrik (2015). « A fast algorithm for reversion of power series ». In : *Mathematics of Computation*, vol. 84, n°291, p. 475–484 (cité page 77).

Jou09 Joux, Antoine (2009). *Algorithmic cryptanalysis*. Chapman & Hall/CRC Cryptography and Network Security Series. Chapman et Hall/CRC (cité pages 13, 434).

JPS01 Jeronimo, Gabriela, Susana Puddu et Juan Sabia (2001). « Computing Chow forms and some applications ». In : *Journal of Algorithms*, vol. 41, n°1, p. 52–68 (cité page 523).

JS00 Jeronimo, Gabriela et Juan Sabia (2000). « Probabilistic equidimensional decomposition ». In : *Comptes Rendus des Séances de l'Académie des Sciences. Série I. Mathématique*, vol. 331, n°6, p. 485–490 (cité page 523).

JS02 — (2002). « Effective equidimensional decomposition of affine varieties ». In : *Journal of Pure and Applied Algebra*, vol. 169, n°2-3, p. 229–248 (cité page 523).

JV05 Jeannerod, Claude-Pierre et Gilles Villard (2005). « Essentially optimal computation of the inverse of generic polynomial matrices ». In : *Journal of Complexity*, vol. 21, n°1, p. 72–86 (cité page 214).

Kal03 Kaltofen, Erich (2003). « Polynomial factorization : a success story ». In : *ISSAC'03 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 3–4 (cité page 412).

Kal59 Kalman, R. E. (1959). « On the general theory of control systems ». In : *IRE Transactions on Automatic Control*, vol. 4, n°3, p. 481–491 (cité page 231).

Kal82a Kaltofen, E. (1982). « A polynomial-time reduction from bivariate to univariate integral polynomial factorization ». In : *23rd Annual Symposium on Foundations of Computer Science (SFCS '08)*. IEEE, p. 57–64 (cité page 412).

Kal82b — (1982). « Polynomial factorization ». In : *Computer algebra. Symbolic and Algebraic Computation*. Éd. par B. Buchberger, G. E. Collins et R. Loos. Springer-Verlag, p. 95–113 (cité page 412).

Kal82c Kaltofen, Erich (1982). « A polynomial reduction from multivariate to bivariate integral polynomial factorization ». In : *STOC'82 : ACM Symposium on Theory of Computing*. ACM Press, p. 261–266 (cité pages 412, 413).

Kal83 — (1983). « On the complexity of finding short vectors in integer lattices ». In : *Computer Algebra : EUROCAL'83, European Computer Algebra Conference London, England, March 28–30, 1983 Proceedings*. Éd. par J. A. Hulzen. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 236–244 (cité page 370).

Kal85a Kaltofen, E. (1985). « Sparse Hensel lifting ». In : *Proceedings of EUROCAL '85, Vol. 2 (Linz, 1985)*. Vol. 204. Lecture Notes in Computer Science. Springer-Verlag, p. 4–17 (cité pages 412, 413).

Kal85b Kaltofen, Erich (1985). « Effective Hilbert irreducibility ». In : *Information and Control*, vol. 66, n°3, p. 123–137 (cité pages 412, 413).

Kal85c — (1985). « Fast parallel absolute irreducibility testing ». In : *Journal of Symbolic Computation*, vol. 1, n°1, p. 57–67 (cité pages 412, 413).

Kal85d — (1985). « Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization ». In : *SIAM Journal on Computing*, vol. 14, n°2, p. 469–489 (cité pages 412, 413).

Kal90 — (1990). « Polynomial factorization 1982–1986 ». In : *Computers in mathematics (Stanford, CA, 1986)*. Vol. 125. Lecture Notes in Pure and Applied Mathematics. Dekker, p. 285–309 (cité page 412).

Kal92 — (1992). « Polynomial factorization 1987–1991 ». In : *LATIN '92 (São Paulo, 1992)*. Vol. 583. Lecture Notes in Computer Science. Springer-Verlag, p. 294–313 (cité page 412).

Kal93a — (1993). « Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems ». In : *Applied algebra, algebraic algorithms and error-correcting codes*. Vol. 673. Lecture Notes in Computer Science. Springer-Verlag, p. 195–212 (cité page 232).

Kal93b — (1993). « Computational differentiation and algebraic complexity theory ». In : *Workshop Report on First Theory Institute on Computational Differentiation*, p. 28–30 (cité page 231).

Kal94 — (1994). « Asymptotically fast solution of Toeplitz-like singular linear systems ». In : *ISSAC'94 : International Symposium on Symbolic and Algebraic Computation*. Oxford, United Kingdom : ACM Press, p. 297–304 (cité page 206).

Kal95a Kaltofen, E. (1995). « Effective Noether irreducibility forms and applications ». In : *Journal of Computer and System Sciences*, vol. 50, n°2, p. 274–295 (cité page 413).

Kal95b Kaltofen, Erich (1995). « Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems ». In : *Mathematics of Computation*, vol. 64, n°210, p. 777–806 (cité page 206).

Kam85 Kaminski, Michael (1985). « A lower bound for polynomial multiplication ». In : *Theoretical Computer Science*, vol. 40, n°2-3, p. 319–322 (cité page 54).

Kap04 Kaporin, I. (2004). « The aggregation and cancellation techniques as a practical tool for faster matrix multiplication ». In : *Theoretical Computer Science*, vol. 315, n°2-3, p. 469–510 (cité page 183).

Kap99 Kaporin, Igor (1999). « A practical algorithm for faster matrix multiplication ». In : *Numerical Linear Algebra with Applications*, vol. 6, n°8, p. 687–700 (cité page 183).

KB70 Knuth, Donald E. et Peter B. Bendix (1970). « Computational problems in abstract algebra ». In : éd. par John Leech. Pergamon Press. Chap. Simple word problems in universal algebras, p. 263–297 (cité page 435).

KCJ00 Kaltofen, E., R. M. Corless et D. J. Jeffrey (2000). « Challenges of symbolic computation : my favorite open problems ». In : *Journal of Symbolic Computation*, vol. 29, n°6, p. 891–919 (cité page 231).

Kel85 Keller-Gehrig, Walter (1985). « Fast algorithms for the characteristic polynomial ». In : *Theoretical Computer Science*, vol. 36, n°2-3, p. 309–317 (cité pages 180, 181, 619).

Kha02 Khan, M. A. H. (2002). « High-order differential approximants ». In : *Journal of Computational and Applied Mathematics*, vol. 149, n°2, p. 457–468 (cité page 154).

KK65 Klyuyev, V. V. et N. I. Kokovkin-Shcherbak (1965). « On the minimization of the number of arithmetic operations for the solution of linear algebraic systems of equations ». In : *USSR Computational Mathematics and Mathematical Physics*, vol. 5, p. 25–43 (cité page 180).

KKB88 Kaminski, M., D. G. Kirkpatrick et N. H. Bshouty (1988). « Addition requirements for matrix and transposed matrix products ». In : *Journal of Algorithms*, vol. 9, n°3, p. 354–364 (cité pages 230, 231).

KKM79a Kailath, T., S. Y. Kung et M. Morf (1979). « Displacement ranks of a matrix ». In : *American Mathematical Society. Bulletin. New Series*, vol. 1, n°5, p. 769–773 (cité page 206).

KKM79b Kailath, Thomas, Sun Yuan Kung et Martin Morf (1979). « Displacement ranks of matrices and linear equations ». In : *Journal of Mathematical Analysis and Applications*, vol. 68, n°2, p. 395–407 (cité page 206).

KM97 Karp, A. H. et P. Markstein (1997). « High-precision division and square root ». In : *ACM Transactions on Mathematical Software*, vol. 23, n°4, p. 561–589 (cité page 77).

Knu71 Knuth, Donald E. (1971). « The analysis of algorithms ». In : *Actes du Congrès International des Mathématiciens 1970*. Vol. 3. Nice : Gauthier-Villars, p. 269–274 (cité page 135).

Knu97 — (1997). *The art of computer programming*. 3e éd. Vol. 2 : Seminumerical Algorithms. Computer Science and Information Processing. Addison-Wesley Publishing Co. (cité pages 12, 135, 342).

KO63 Karatsuba, A. et Y. Ofman (1963). « Multiplication of multidigit numbers on automata ». In : *Soviet Physics Doklady*, vol. 7, p. 595–596 (cité page 52).

Kou10 Koutschan, Christoph (2010). « A fast approach to creative telescoping ». In : *Mathematics in Computer Science*, vol. 4, n°2-3, p. 259–266 (cité page 588).

KP11 Kauers, Manuel et Peter Paule (2011). *The concrete tetrahedron*. Texts and Monographs in Symbolic Computation. Symbolic sums, recurrence equations, generating functions, asymptotic estimates. Springer, Vienna (cité page 541).

KP81 Knuth, Donald E. et Christos H. Papadimitriou (1981). « Duality in addition chains ». In : *Bulletin of the European Association for Theoretical Computer Science*, vol. 13, p. 2–4 (cité pages 231, 232).

KP96 Krick, T. et L. M. Pardo (1996). « A computational method for Diophantine approximation ». In : *Algorithms in algebraic geometry and applications (Santander, 1994)*. Vol. 143. Progress in Mathematics. Birkhäuser, p. 193–253 (cité page 522).

Kro82 Kronecker, Leopold (1882). « Grundzüge einer arithmetischen Theorie der algebraischen Grössen ». In : *Journal für die Reine und Angewandte Mathematik*, vol. 92, p. 1–122 (cité pages 464, 522).

Kry31 Krylov, A. N. (1931). « On the numerical solution of the equation by which in technical questions frequencies of small oscillations of material systems are determined ». In : *Izvestiya Akademii Nauk SSSR*, vol. 7, n°4. (in Russian), p. 491–539 (cité page 181).

KS01 Koy, Henrik et Claus Peter Schnorr (2001). « Segment LLL-Reduction of Lattice Bases ». In : *Cryptography and Lattices : International Conference, CaLC 2001 Providence, RI, USA, March 29–30, 2001 Revised Papers*. Éd. par Joseph H. Silverman. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 67–80 (cité page 370).

KS73 Kogge, P. et H. Stone (1973). « A parallel algorithm for the efficient solution of a general class of recurrence equations ». In : *IEEE Transactions on Computers*, vol. C-22, p. 786–793 (cité page 322).

KS91 Kaltofen, Erich et B. David Saunders (1991). « On Wiedemann's method of solving sparse linear systems ». In : *Applied algebra, algebraic algorithms and error-correcting codes (New Orleans, LA, 1991)*. Vol. 539. Lecture Notes in Computer Science. Springer-Verlag, p. 29–38 (cité page 193).

KS98 Kaltofen, Erich et Victor Shoup (1998). « Subquadratic-time factoring of polynomials over finite fields ». In : *Mathematics of Computation*, vol. 67, n°223, p. 1179–1197 (cité page 356).

KT77 Kung, H. T. et D. M. Tong (1977). « Fast algorithms for partial fraction decomposition ». In : *SIAM Journal on Computing*, vol. 6, n°3, p. 582–593 (cité page 619).

KU08 Kedlaya, Kiran S. et Christopher Umans (2008). « Fast modular composition in any characteristic ». In : *FOCS'08 : IEEE Conference on Foundations of Computer Science*. Washington, DC, USA : IEEE Computer Society, p. 146–155 (cité pages 77, 356).

Kun74 Kung, H. T. (1974). « On computing reciprocals of power series ». In : *Numerische Mathematik*, vol. 22, p. 341–348 (cité page 77).

KV04 Kaltofen, Erich et Gilles Villard (2004). « On the complexity of computing determinants ». In : *Computational Complexity*, vol. 13, n°3-4, p. 91–130 (cité page 214).

Lad76 Laderman, Julian D. (1976). « A noncommutative algorithm for multiplying 3×3 matrices using 23 muliplications ». In : *Bulletin of the American Mathematical Society*, vol. 82, n°1, p. 126–128 (cité page 182).

Lag68 Lagrange, Joseph Louis, comte de (1768). « Nouvelle méthode pour résoudre les équations littérales par le moyen des séries ». In : *Mémoires de l'Académie Royale des Sciences et Belles-Lettres de Berlin*, vol. 24, p. 251–326 (cité page 77).

Lag73 — (1773). « Recherches d'arithmétique ». In : *Nouveaux Mémoires de l'Académie Royale des Sciences et Belles-Lettres de Berlin*, p. 265–312 (cité page 370).

Lag75 — (1775). « Recherches d'arithmétique ». In : *Nouveaux Mémoires de l'Académie Royale des Sciences et Belles-Lettres de Berlin*, p. 323–356 (cité page 370).

Lak90 Lakshman, Y. N. (1990). « On the complexity of computing a Gröbner basis for the radical of a zero dimensional ideal ». In : *Proceedings of the Twenty-second Annual*

ACM Symposium on Theory of Computing. STOC '90. New York, NY, USA : ACM, p. 555–563 (cité page 481).

Lak91 Lakshman, Y. N. (1991). « A single exponential bound on the complexity of computing Gröbner bases of zero dimensional ideals ». In : *Effective Methods in Algebraic Geometry*. Éd. par Teo Mora et Carlo Traverso. Boston, MA : Birkhäuser Boston, p. 227–234 (cité page 481).

Lan02 Lang, Serge (2002). *Algebra*. 3e éd. Vol. 211. Graduate Texts in Mathematics. Springer-Verlag (cité pages 12, 135).

Lan05 Landau, E. (1905). « Sur quelques théorèmes de M. Petrovitch relatifs aux zéros des fonctions analytiques ». In : *Bulletin de la S. M. F.* vol. 33, p. 251–261 (cité page 396).

Lan06 Landsberg, J. M. (2006). « The border rank of the multiplication of 2×2 matrices is seven ». In : *Journal of the American Mathematical Society*, vol. 19, n°2, p. 447–459 (cité page 182).

Lan08 — (2008). « Geometry and the complexity of matrix multiplication ». In : *Bulletin of the American Mathematical Society*, vol. 45, n°2, p. 247–284 (cité page 184).

Lan14 — (2014). « New lower bounds for the rank of matrix multiplication ». In : *SIAM Journal on Computing*, vol. 43, n°1, p. 144–149 (cité page 182).

Las03 Lascoux, Alain (2003). *Symmetric functions and combinatorial operators on polynomials*. Vol. 99. CBMS Regional Conference Series in Mathematics. Published for the Conference Board of the Mathematical Sciences, Washington, DC (cité page 135).

Laz83 Lazard, D. (1983). « Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations ». In : *Computer Algebra : EUROCAL'83, European Computer Algebra Conference London, England, March 28–30, 1983 Proceedings*. Éd. par J. A. Hulzen. Springer Berlin Heidelberg, p. 146–156 (cité pages 444, 481).

Le 14 Le Gall, François (2014). « Powers of tensors and fast matrix multiplication ». In : *ISSAC'14 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Katsusuke Nabeshima. New York, NY, USA : ACM, p. 296–303 (cité page 183).

Le 40 Le Verrier, Urbain (1840). « Sur les variations séculaires des éléments elliptiques des sept planètes principales : Mercure, Vénus, la Terre, Mars, Jupiter, Saturne et Uranus ». In : *Journal de Mathématiques Pures et Appliquées*, vol. 1 (5), p. 220–254 (cité page 181).

Lec00 Lecerf, Grégoire (2000). « Computing an equidimensional decomposition of an algebraic variety by means of geometric resolutions ». In : *ISSAC'00 : International Symposium on Symbolic and Algebraic Computation*. ACM, p. 209–216 (cité page 523).

Lec02 — (2002). « Quadratic Newton iteration for systems with multiplicity ». In : *Foundations of Computational Mathematics*, vol. 2, n°3, p. 247–293 (cité page 523).

Lec03 — (2003). « Computing the equidimensional decomposition of an algebraic closed set by means of lifting fibers ». In : *Journal of Complexity*, vol. 19, n°4, p. 564–596 (cité page 523).

Lec06 — (2006). « Sharp precision in Hensel lifting for bivariate polynomial factorization ». In : *Mathematics of Computation*, vol. 75, p. 921–933 (cité page 413).

Lec07 — (2007). « Improved dense multivariate polynomial factorization algorithms ». In : *Journal of Symbolic Computation*, vol. 42, n°4, p. 477–494 (cité page 413).

Lec08 — (2008). « Fast separable factorization and applications ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 19, n°2, p. 135–160 (cité pages 342, 396, 412).

Lec10 — (2010). « New recombination algorithms for bivariate polynomial factorization based on Hensel lifting ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 21, n°2, p. 151–176 (cité page 413).

Lec13 — (2013). « Factorisation des polynômes à plusieurs variables (Cours no. II) ». In : *Journées nationales de calcul formel*. Éd. par G. Chèze, P. Boito, C. Pernet et M. Safey el Din. Vol. 3. Les cours du CIRM n°1. Cedram, p. 1–85. url : http://ccirm.cedram.org/ccirm-bin/fitem?id=CCIRM_2013__3_1_A2_0 (cité pages 342, 413).

Leh38 Lehmer, D. H. (1938). « Euclid's Algorithm for large numbers ». In : *The American Mathematical Monthly*, vol. 45, n°4, p. 227–233 (cité page 135).

Lej84 Lejeune-Jalabert, Monique (1984). *Effectivité de calculs polynomiaux*. Vol. 19. Cours de l'Institut Fourier. Cours de DEA 84–85. Université de Grenoble I. url : http://www.numdam.org/numdam-bin/feuilleter?id=CIF_1984-1985__19_ (cité page 481).

Lev47 Levinson, Norman (1947). « The Wiener RMS (root mean square) error criterion in filter design and prediction ». In : *Journal of Mathematics and Physics*, vol. 25, p. 261–278 (cité page 206).

Li00 Li, Ziming (2000). « Greatest common right divisors, least common left multiples, and subresultants of Ore polynomials ». In : *Mathematics mechanization and applications*. San Diego, CA : Academic Press, p. 297–324 (cité page 558).

Li98 — (1998). « A subresultant theory for Ore polynomials with applications ». In : *ISSAC'98 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 132–139 (cité page 558).

Lic87 Lickteig, Thomas (1987). « The computational complexity of division in quadratic extension fields ». In : *SIAM Journal on Computing*, vol. 16, n°2, p. 278–311 (cité page 180).

Lio33 Liouville, Joseph (1833). « Second mémoire sur la détermination des intégrales dont la valeur est algébrique ». In : *Journal de l'École polytechnique*, vol. 14, p. 149–193 (cité page 321).

Lip76 Lipson, J. D. (1976). « Newton's method : a great *algebraic* algorithm ». In : *SYMSAC'76 : ACM Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 260–270 (cité page 77).

Lip89 Lipshitz, L. (1989). « D-finite power series ». In : *Journal of Algebra*, vol. 122, n°2, p. 353–373 (cité page 266).

LL91 Lakshman, Y. N. et Daniel Lazard (1991). « On the complexity of zero-dimensional algebraic systems ». In : *Effective Methods in Algebraic Geometry*. Éd. par Teo Mora et Carlo Traverso. Boston, MA : Birkhäuser Boston, p. 217–225 (cité page 481).

LLL82 Lenstra, A. K., H. W. Lenstra Jr. et L. Lovász (1982). « Factoring polynomials with rational coefficients ». In : *Mathematische Annalen*, vol. 261, n°4, p. 515–534 (cité pages 370, 396).

LN97 Li, Z. et I. Nemes (1997). « A modular algorithm for computing greatest common right divisors of Ore polynomials ». In : *ISSAC'97 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 282–289 (cité page 558).

LPS92 Laderman, Julian, Victor Pan et Xuan He Sha (1992). « On practical algorithms for accelerated matrix multiplication ». In : *Linear Algebra and its Applications*, vol. 162/164, p. 557–588 (cité page 183).

LS03 Lecerf, G. et É. Schost (2003). « Fast multivariate power series multiplication in characteristic zero ». In : *SADIO Electronic Journal on Informatics and Operations Research*, vol. 5, n°1, p. 1–10 (cité page 231).

Mac94 Macaulay, F. S. (1994). *The algebraic theory of modular systems*. Cambridge Mathematical Library. Revised reprint of the 1916 original, With an introduction by Paul Roberts. Cambridge University Press (cité page 481).

Mah61 Mahler, K. (1961). « On the zeros of the derivative of a polynomial ». In : *Proceedings of the Royal Society of London A : Mathematical, Physical and Engineering Sciences*, vol. 264, n°1317, p. 145–154 (cité page 396).

Mah68 — (1968). « Perfect systems ». In : *Compositio Mathematica*, vol. 19, 95–166 (1968) (cité page 154).

Mak86 Makarov, O. M. (1986). « An algorithm for multiplying 3×3 matrices ». In : *USSR Computational Mathematics and Mathematical Physics*, vol. 26, n°1, p. 179–180 (cité page 182).

Mar98 Marotte, F. (1898). « Les équations différentielles linéaires et la théorie des groupes ». In : *Annales de la Faculté des Sciences de Toulouse*, vol. 12, n°4, p. 1–92 (cité page 321).

Mas69 Massey, James L. (1969). « Shift-register synthesis and BCH decoding ». In : *IEEE Transactions on Information Theory*, vol. IT-15, p. 122–127 (cité page 153).

Mat93 Matiyasevich, Yuri V. (1993). *Hilbert's tenth problem*. Foundations of Computing Series. Translated from the 1993 Russian original by the author, with a foreword by Martin Davis. MIT Press (cité page 30).

MB66 Miller, J. C. P. et D. J. Spencer Brown (1966). « An algorithm for evaluation of remote terms in a linear recurrence sequence ». In : *Computer Journal*, vol. 9, p. 188–190 (cité page 92).

MB72 Moenck, R. T. et A. Borodin (1972). « Fast modular transforms via division ». In : *Thirteenth Annual IEEE Symposium on Switching and Automata Theory*, p. 90–96 (cité pages 91, 107).

MC79 Moenck, Robert T. et John H. Carter (1979). « Approximate algorithms to derive exact solutions to systems of linear equations ». In : *Symbolic and Algebraic Computation. Eurosam '79, An International Symposium on Symbolic and Algebraic Manipulation, Marseille, France, June 1979*. Vol. 72. Lecture Notes in Computer Science. London, UK : Springer-Verlag, p. 65–73 (cité page 214).

Mer74 Mersereau, Russell M. (1974). « An algorithm for performing an inverse chirp z-transform ». In : *IEEE Transactions on Audio and Electroacoustics*, vol. ASSP-22, n°5, p. 387–388 (cité page 107).

Mig74 Mignotte, M. (1974). « An inequality about factors of polynomials ». In : *Mathematics of Computation*, vol. 28, p. 1153–1157 (cité page 396).

Mil75 Mills, W. H. (1975). « Continued fractions and linear recurrences ». In : *Mathematics of Computation*, vol. 29, p. 173–180 (cité page 153).

MM82 Mayr, Ernst W. et Albert R. Meyer (1982). « The complexity of the word problems for commutative semigroups and polynomial ideals ». In : *Advances in Mathematics*, vol. 46, n°3, p. 305–329 (cité page 481).

MM86 Möller, H. Michael et Ferdinando Mora (1986). « New constructive methods in classical ideal theory ». In : *Journal of Algebra*, vol. 100, n°1, p. 138–178 (cité page 444).

Moe73 Moenck, R. T. (1973). « Fast computation of GCDs ». In : *STOC'73 : ACM Symposium on Theory of Computing*. Austin : ACM, p. 142–151 (cité page 135).

Moe76 Moenck, Robert T. (1976). « Another polynomial homomorphism ». In : *Acta Informatica*, vol. 6, n°2, p. 153–169 (cité page 54).

Moe77 — (1977). « On the efficiency of algorithms for polynomial factoring ». In : *Mathematics of Computation*, vol. 31, p. 235–250 (cité page 356).

Möl08 Möller, Niels (2008). « On Schönhage's algorithm and subquadratic integer GCD computation ». In : *Mathematics of Computation*, vol. 77, n°261, p. 589–607 (cité page 136).

Mon05 Montgomery, Peter L. (2005). « Five, six, and seven-term Karatsuba-like formulae ». In : *IEEE Transactions on Computers*, vol. 54, n°3, p. 362–369 (cité pages 53, 54).

Mon85 — (1985). « Modular multiplication without trial division ». In : *Mathematics of Computation*, vol. 44, n°170, p. 519–521 (cité page 91).

Mon92 Montgomery, P. L. (1992). « An FFT extension of the elliptic curve method of factorization ». Thèse de doctorat. University of California, Los Angeles CA (cité page 107).

Mor05 Mora, Teo (2005). *Solving polynomial equation systems II : Macaulay's paradigm and Gröbner technology*. Vol. 99. Encyclopedia of Mathematics and its Applications. Cambridge University Press (cité page 444).

Mor73 Morgenstern, Jacques (1973). « Note on a lower bound of the linear complexity of the fast Fourier transform ». In : *Journal of the Association for Computing Machinery*, vol. 20, p. 305–306 (cité page 53).

Mor80 Morf, M. (1980). « Doubling algorithms for Toeplitz and related equations ». In : *IEEE Conference on Acoustics, Speech, and Signal Processing*, p. 954–959 (cité page 206).

MP13 Mullen, Gary L. et Daniel Panario (2013). *Handbook of finite fields*. Discrete Mathematics and Its Applications. Chapman et Hall/CRC (cité pages 356, 413).

MRR88 Mines, Ray, Fred Richman et Wim Ruitenburg (1988). *A course in constructive algebra*. Universitext. Springer-Verlag (cité page 342).

MS10 Mezzarobba, Marc et Bruno Salvy (2010). « Effective Bounds for P-Recursive Sequences ». In : *Journal of Symbolic Computation*, vol. 45, n°10, p. 1075–1096 (cité page 322).

MS78 McEliece, Robert J. et James B. Shearer (1978). « A property of Euclid's algorithm and an application to Padé approximation ». In : *SIAM Journal on Applied Mathematics*, vol. 34, n°4, p. 611–615 (cité page 153).

MS88 Mignotte, Maurice et Claus Schnorr (1988). « Calcul déterministe des racines d'un polynôme dans un corps fini ». In : *Comptes Rendus des Séances de l'Académie des Sciences. Série I. Mathématique*, vol. 306, n°12, p. 467–472 (cité page 356).

Mul00 Mulders, Thom (2000). « On short multiplications and divisions ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 11, n°1, p. 69–88 (cité page 54).

Mul04 — (2004). « Certified sparse linear system solving ». In : *Journal of Symbolic Computation*, vol. 38, n°5, p. 1343–1373 (cité page 193).

Mun73 Munro, I. (1973). « Problems related to matrix multiplication ». In : *Proceedings Courant Institute Symposium on Computational Complexity, October 1971*. Éd. par R. Rustin. New York : Algorithmics Press, p. 137–151 (cité page 180).

Mus71 Musser, David R. (1971). « Algorithms for polynomial factorization ». Thèse de doctorat. Univ. of Wisconsin : C.S. Department (cité page 342).

Mus75 — (1975). « Multivariate polynomial factorization ». In : *Journal of the ACM*, vol. 22, p. 291–308 (cité page 412).

MW96 Macintyre, Angus et A. J. Wilkie (1996). « On the decidability of the real exponential field ». In : *Kreiseliana*. A. K. Peters, p. 441–467 (cité page 30).

New40 Newton, Isaac (1740). *La méthode des fluxions, et les suites infinies*. Traduction française par G. Buffon du texte de 1671 de Newton. de Bure. url : http://www.gallica.fr (cité page 76).

NH09 Novocin, Andy et Mark van Hoeij (2009). « Factoring univariate polynomials over the rationals ». In : *ACM Communications in Computer Algebra*, vol. 42, n°3, p. 157–157 (cité page 397).

Nov08 Novocin, Andrew (2008). « Factoring univariate polynomials over the rationals ». Thèse de doctorat. Department of Mathematics, Florida State University, Tallahassee (cité page 397).

NS05 Nguyên, Phong Q. et Damien Stehlé (2005). « Floating-point LLL revisited ». In : *Proceedings of Advances in Cryptology, EUROCRYPT 2005 : 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005*. Éd. par Ronald Cramer. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 215–233 (cité page 370).

NS09 Nguyen, Phong Q. et Damien Stehlé (2009). « An LLL algorithm with quadratic complexity ». In : *SIAM Journal on Computing*, vol. 39, n°3, p. 874–903 (cité page 370).

NSV11 Novocin, Andrew, Damien Stehlé et Gilles Villard (2011). « An LLL-reduction algorithm with quasi-linear time complexity ». In : *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*. STOC '11. Extended abstract. New York, NY, USA : ACM, p. 403–412 (cités pages 370, 397).

Nus80 Nussbaumer, Henri J. (1980). « Fast polynomial transform algorithms for digital convolution ». In : *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, n°2, p. 205–215 (cité page 53).

Nus81 — (1981). *Fast Fourier transform and convolution algorithms*. Vol. 2. Springer Series in Information Sciences. Springer-Verlag (cité page 52).

NW78 Nijenhuis, Albert et Herbert S. Wilf (1978). *Combinatorial algorithms for computers and calculators*. Academic Press (cité page 396).

Olv+10 Olver, Frank W. J., Daniel W. Lozier, Ronald F. Boisvert et Charles W. Clark, éds. (2010). *NIST handbook of mathematical functions*. U.S. Department of Commerce, National Institute of Standards et Technology, Washington, DC ; Cambridge University Press, Cambridge (cité page 266).

Ore31 Ore, Oystein (1931). « Linear equations in non-commutative fields ». In : *Annals of Mathematics*, vol. 32, p. 463–477 (cité page 558).

Ore33 — (1933). « Theory of non-commutative polynomials ». In : *Annals of Mathematics*, vol. 34, n°3, p. 480–508 (cité page 558).

Ost54 Ostrowski, A. M. (1954). « On two problems in abstract algebra connected with Horner's rule ». In : *Studies in mathematics and mechanics presented to Richard von Mises*. NY : Academic Press Inc., p. 40–48 (cité page 107).

Pad92 Padé, H. (1892). « Sur la représentation approchée d'une fonction par des fractions rationelles ». In : *Annales scientifiques de l'É.N.S.* vol. 9, p. 3–93 (cité page 154).

Pad94 — (1894). « Sur la généralisation des fractions continues algébriques ». In : *Journal de mathématiques pures et appliquées*, vol. 10, n°4, p. 291–330 (cité page 154).

Pan01 Pan, Victor Y. (2001). *Structured matrices and polynomials*. Unified superfast algorithms. Birkhäuser Boston Inc. (cités pages 13, 206).

Pan66 Pan, V. Ya. (1966). « Methods of computing values of polynomials ». In : *Russian Mathematical Surveys*, vol. 21, n°1, p. 105–136 (cité page 107).

Pan72 Pan, V. (1972). « Computation schemes for a product of matrices and for the inverse matrix ». In : *Akademiya Nauk SSSR i Moskovskoe Matematicheskoe Obshchestvo. Uspekhi Matematicheskikh Nauk*, vol. 27, n°5(167), p. 249–250 (cité page 183).

Pan78 Pan, V. Ya. (1978). « Strassen's algorithm is not optimal. Trilinear technique of aggregating, uniting and canceling for constructing fast algorithms for matrix operations ». In : *SFCS '78*. Washington, DC, USA : IEEE Computer Society, p. 166–176 (cité page 182).

Pan80 — (1980). « New fast algorithms for matrix operations ». In : *SIAM Journal on Computing*, vol. 9, n°2, p. 321–342 (cité page 182).

Pan81 — (1981). « New combinations of methods for the acceleration of matrix multiplication ». In : *Computers & Mathematics with Applications. An International Journal*, vol. 7, n°1, p. 73–125 (cité page 183).

Pan84a Pan, Victor (1984). « How can we speed up matrix multiplication ? » In : *SIAM Review*, vol. 26, n°3, p. 393–415 (cité page 180).

Pan84b — (1984). *How to multiply matrices faster*. Vol. 179. Lecture Notes in Computer Science. Springer-Verlag (cité pages 13, 180).

Pan90 — (1990). « On computations with dense structured matrices ». In : *Mathematics of Computation*, vol. 55, n°191, p. 179–190 (cité page 206).

Pas87 Paszkowski, Stefan (1987). « Recurrence relations in Padé–Hermite approximation ». In : *Journal of Computational and Applied Mathematics*, vol. 19, n°1, p. 99–107 (cité page 154).

Pat75 Paterson, Michael S. (1975). « Complexity of monotone networks for Boolean matrix product ». In : *Theoretical Computer Science*, vol. 1, n°1, p. 13–20 (cité page 180).

Pet92 Petkovšek, Marko (1992). « Hypergeometric solutions of linear recurrences with polynomial coefficients ». In : *Journal of Symbolic Computation*, vol. 14, n°2-3, p. 243–264 (cité page 542).

PFM74 Paterson, M. S., M. J. Fischer et A. R. Meyer (1974). « An improved overlap argument for on-line multiplication ». In : *Complexity of computation*. AMS, p. 97–111 (cité page 53).

Pic05 Picard, Émile, éd. (1905). *Œuvres de Charles Hermite*. Gauthier-Villars (cité page 370).

Poo89 Poorten, A. J. van der (1989). « Some facts that should be better known, especially about rational functions ». In : *Number theory and applications*. Proceedings of a Conference held at Banff, AB, 1988. Dordrecht : Kluwer, p. 497–528 (cité page 91).

Pro76 Probert, Robert L. (1976). « On the additive complexity of matrix multiplication ». In : *SIAM Journal on Computing*, vol. 5, n°2, p. 187–203 (cité page 182).

PS03 Put, Marius van der et Michael F. Singer (2003). *Galois theory of linear differential equations*. Vol. 328. Grundlehren der Mathematischen Wissenschaften. Springer-Verlag (cité page 558).

PS07 Pernet, Clément et Arne Storjohann (2007). « Faster algorithms for the characteristic polynomial ». In : *ISSAC'07 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Dongming Wang. ACM Press, p. 307–314 (cité page 181).

PS12 Pauderis, Colton et Arne Storjohann (2012). « Deterministic unimodularity certification ». In : *ISSAC 2012—Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation*. ACM, New York, p. 281–288 (cité page 214).

PS73 Paterson, M. S. et L. J. Stockmeyer (1973). « On the number of nonscalar multiplications necessary to evaluate polynomials ». In : *SIAM Journal on Computing*, vol. 2, n°1, p. 60–66 (cité pages 77, 181).

PS97 Put, Marius van der et Michael F. Singer (1997). *Galois theory of difference equations*. Vol. 1666. Lecture Notes in Mathematics. Springer-Verlag (cité page 558).

PSD70 Penfield Jr., P., R. Spence et S. Duinker (1970). *Tellegen's theorem and electrical networks*. The M.I.T. Press, Cambridge, Mass.-London (cité page 231).

PW02 Pan, V. Y. et X. Wang (2002). « Acceleration of Euclidean algorithm and extensions ». In : *ISSAC'02 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Teo Mora. Lille : ACM Press, p. 207–213 (cité page 136).

PW04 Pan, Victor Y. et Xinmao Wang (2004). « On rational number reconstruction and approximation ». In : *SIAM Journal on Computing*, vol. 33, n°2, p. 502–503 (cité page 153).

PWZ96 Petkovšek, Marko, Herbert S. Wilf et Doron Zeilberger (1996). A = B. A. K. Peters (cité pages 13, 541).

PZ00 Pan, Victor Y. et Ailong Zheng (2000). « Superfast algorithms for Cauchy-like matrix computations and extensions ». In : *Linear Algebra and its Applications*, vol. 310, n°1-3, p. 83–108 (cité page 206).

Raz03 Raz, Ran (2003). « On the complexity of matrix product ». In : *SIAM Journal on Computing*, vol. 32, n°5, p. 1356–1369 (cité page 182).

Ric68 Richardson, Daniel (1968). « Some undecidable problems involving elementary functions of a real variable ». In : *Journal of Symbolic Logic*, vol. 33, n°4, p. 514–520 (cité page 30).

Ric69 — (1969). « Solution of the identity problem for integral exponential functions ». In : *Zeitschrift fur mathematischen Logik und Grundlagen der Mathematik*, vol. 15, p. 333–340 (cité page 30).

Ric97 — (1997). « How to recognize zero ». In : *Journal of Symbolic Computation*, vol. 24, n°6, p. 627–645 (cité page 30).

Rit86 Ritzmann, P. (1986). « A fast numerical algorithm for the composition of power series with complex coefficients ». In : *Theoretical Computer Science*, vol. 44, p. 1–16 (cité page 77).

Rob85 Robbiano, Lorenzo (1985). « Term orderings on the polynomial ring ». In : *EUROCAL '85. European Conference on Computer Algebra. Linz, Austria, April 1-3, 1985. Proceedings. Volume 2 : Research Contributions*. Éd. par Bob F. Caviness. Vol. 204. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, p. 513–517 (cité page 434).

Rom84 Roman, Steven (1984). *The umbral calculus*. Vol. 111. Pure and Applied Mathematics. Academic Press Inc. [Harcourt Brace Jovanovich Publishers] (cité page 231).

Rón89 Rónyai, L. (1989). « Factoring polynomials modulo special primes ». In : *Combinatorica. An International Journal on Combinatorics and the Theory of Computing*, vol. 9, n°2, p. 199–206 (cité page 356).

Rón92 Rónyai, Lajos (1992). « Galois groups and factoring polynomials over finite fields ». In : *SIAM Journal on Discrete Mathematics*, vol. 5, n°3, p. 345–365 (cité page 356).

Rou01 Roura, Salvador (2001). « Improved master theorems for divide-and-conquer recurrences ». In : *Journal of the Association for Computing Machinery*, vol. 48, n°2, p. 170–205 (cité page 30).

Rou99 Rouillier, Fabrice (1999). « Solving zero-dimensional systems through the rational univariate representation ». In : *Applicable Algebra in Engineering, Communication and Computing*, vol. 9, n°5, p. 433–461 (cité page 464).

RR90 Risler, Jean-Jacques et Felice Ronga (1990). « Testing polynomials ». In : *Journal of Symbolic Computation*, vol. 10, n°1, p. 1–5 (cité page 523).

RSR69 Rabiner, L. R., R. W. Schafer et C. M. Rader (1969). « The chirp z-transform algorithm and its application ». In : *Bell System Technical Journal*, vol. 48, p. 1249–1292 (cité page 107).

Rup86 Ruppert, Wolfgang M. (1986). « Reduzibilität ebener Kurven ». In : *Journal für die reine und angewandte Mathematik*, vol. 369, p. 167–191 (cité page 412).

Rup99 — (1999). « Reducibility of polynomials $f(x, y)$ modulo p ». In : *Journal of Number Theory*, vol. 77, n°1, p. 62–70 (cité page 412).

Sch00 Schinzel, Andrzej (2000). *Polynomials with special regard to reducibility*. Vol. 77. Encyclopedia of Mathematics and its Applications. Cambridge University Press (cité page 413).

Sch01 Schost, Éric (2001). « Sur la résolution des systèmes polynomiaux à paramètres ». Thèse de doctorat. Palaiseau, France : École polytechnique (cité page 522).

Sch03 Schnorr, Claus Peter (2003). « Lattice reduction by random sampling and birthday methods ». In : *STACS 2003 : 20th Annual Symposium on Theoretical Aspects of Computer Science Berlin, Germany, February 27 – March 1, 2003 Proceedings*. Éd. par Helmut Alt et Michel Habib. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 145–156 (cité page 370).

Sch33 Schulz, G. (1933). « Iterative Berechnung der reziproken Matrix ». In : *Zeitschrift für angewandte Mathematik und Physik*, vol. 13, p. 57–59 (cité page 77).

Sch71 Schönhage, A. (1971). « Schnelle Berechnung von Kettenbruchentwicklugen ». In : *Acta Informatica*, vol. 1, p. 139–144 (cité page 135).

Sch73 — (1972/73). « Unitäre Transformationen grosser Matrizen ». In : *Numerische Mathematik*, vol. 20, p. 409–417 (cité page 180).

Sch77 — (1976/77). « Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2 ». In : *Acta Informatica*, vol. 7, n°4, p. 395–398 (cité page 53).

Sch80a Schreyer, Frank-Olaf (1980). « Die Berechnung Syzygien mit dem verallgemeinerten Weierstrasschen Divisionsatz ». Diplomarbeit. Hamburg (cité page 444).

Sch80b Schwartz, J. T. (1980). « Fast probabilistic algorithms for verification of polynomial identities ». In : *Journal of the Association for Computing Machinery*, vol. 27, n°4, p. 701–717 (cité pages 136, 193).

Sch81 Schönhage, A. (1981). « Partial and total matrix multiplication ». In : *SIAM Journal on Computing*, vol. 10, n°3, p. 434–455 (cité pages 181, 183).

Sch82 — (1982). *The fundamental theorem of algebra in terms of computational complexity*. Preliminary Report. 73 pages. Tübingen, Germany : Mathematisches Institut der Universität (cité page 77).

Sch84 Schönhage, Arnold (1984). « Factorization of univariate integer polynomials by diophantine approximation and an improved basis reduction algorithm ». In : *Automata, Languages and Programming : 11th Colloquium Antwerp, Belgium, July 16–20, 1984*. Éd. par Jan Paredaens. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 436–447 (cité pages 370, 396).

Sch85 Schoof, René (1985). « Elliptic curves over finite fields and the computation of square roots mod p ». In : *Mathematics of Computation*, vol. 44, n°170, p. 483–494 (cité page 356).

Sch87 Schnorr, C. P. (1987). « A hierarchy of polynomial time lattice basis reduction algorithms ». In : *Theoretical Computer Science*, vol. 53, n°2-3, p. 201–224 (cité page 370).

Sch88a — (1988). « A more efficient algorithm for lattice basis reduction ». In : *Journal of Algorithms*, vol. 9, n°1, p. 47–62 (cité page 370).

Sch88b Schönhage, A. (1988). « Probabilistic computation of integer polynomial GCDs ». In : *Journal of Algorithms*, vol. 9, n°3, p. 365–371 (cité page 136).

SE94 Schnorr, C. P. et M. Euchner (1994). « Lattice basis reduction : improved practical algorithms and solving subset sum problems ». In : *Mathematical Programming*, vol. 66, n°1, p. 181–199 (cité page 370).

Ser86 Sergeyev, A. V. (1986). « A recursive algorithm for Padé–Hermite approximations ». In : *USSR Comput. Maths Math. Phys*, vol. 26, n°2, p. 17–22 (cité page 154).

Sha74 Shafer, R. E. (1974). « On quadratic approximation ». In : *SIAM Journal on Numerical Analysis*, vol. 11, p. 447–460 (cité page 154).

Sha94 Shafarevich, I. R. (1994). *Basic algebraic geometry. 1 Varieties in projective space*. 2e éd. Springer-Verlag (cité page 413).

Sho09 Shoup, Victor (2009). *A computational introduction to number theory and algebra*. 2e éd. Cambridge University Press (cité pages 13, 356).

Sho90 — (1990). « On the deterministic complexity of factoring polynomials over finite fields ». In : *Information Processing Letters*, vol. 33, n°5, p. 261–267 (cité page 356).

Sho91a Shoup, V. (1991). « A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic ». In : *ISSAC'91 : International Symposium on Symbolic and Algebraic Computation*. ACM Press, p. 14–21 (cité pages 92, 230, 231).

Sho91b Shoup, Victor (1991). « A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic ». In : *ISSAC'91 : International Symposium on Symbolic and Algebraic Computation*. Éd. par S. M. Watt. ACM Press, p. 14–21 (cité page 356).

Sho94 — (1994). « Fast construction of irreducible polynomials over finite fields ». In : *Journal of Symbolic Computation*, vol. 17, n°5, p. 371–391 (cité page 231).

Sho95 Shoup, V. (1995). « A new polynomial factorization algorithm and its implementation ». In : *Journal of Symbolic Computation*, vol. 20, n°4, p. 363–397 (cité page 231).

Sho99 — (1999). « Efficient computation of minimal polynomials in algebraic extensions of finite fields ». In : *ISSAC'99 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 53–58 (cité pages 230, 231).

Sie72 Sieveking, M. (1972). « An algorithm for division of powerseries ». In : *Computing*, vol. 10, p. 153–156 (cité page 77).

Sin07 Singer, M. F. (2007). « Introduction to the Galois theory of linear differential equations ». url : https://arxiv.org/abs/0712.4124 (cité page 558).

Sin81 Singer, Michael F. (1981). « Liouvillian solutions of n-th order homogeneous linear differential equations ». In : *American Journal of Mathematics*, vol. 103, n°4, p. 661–682 (cité page 321).

Sin86 — (1986). « Algebraic relations among solutions of linear differential equations ». In : *Transactions of the American Mathematical Society*, vol. 295, n°2, p. 753–763 (cité page 267).

Slo06 Sloane, N. J. A. (2006). *The on-line encyclopedia of integer sequences*. url : http://www.research.att.com/~njas/sequences/ (cité page 266).

Smi02 Smith, Warren D. (2002). « Fast matrix multiplication formulae : report of the prospectors ». Preprint. http://www.math.temple.edu/~wds/prospector.pdf (cité pages 182, 183).

Smi62 Smith, R. L. (1962). « Algorithm 116 : Complex division ». In : *Communications of the ACM*, n°5, p. 435 (cité page 180).

SP95 Sloane, N. J. A. et Simon Plouffe (1995). *The encyclopedia of integer sequences*. Academic Press, Inc., San Diego, CA (cité page 266).

Spe78 Spear, D. (1978). « A constructive approach to commutative ring theory ». In : *Processing of the 1977 MACSYMA Users' Conference (NASA)*, p. 369–376 (cité page 444).

SS71 Schönhage, A. et V. Strassen (1971). « Schnelle Multiplikation großer Zahlen ». In : *Computing*, vol. 7, p. 281–292 (cité page 53).

SS97 Shoup, Victor et Roman Smolensky (1996/97). « Lower bounds for polynomial evaluation and interpolation problems ». In : *Computational Complexity*, vol. 6, n°4, p. 301–311 (cité page 107).

SST00 Saito, Mutsumi, Bernd Sturmfels et Nobuki Takayama (2000). *Gröbner deformations of hypergeometric differential equations*. Springer-Verlag (cité pages 13, 576).

Sta80 Stanley, R. P. (1980). « Differentiably finite power series ». In : *European Journal of Combinatorics*, vol. 1, n°2, p. 175–188 (cité page 266).

Sta99 Stanley, Richard P. (1999). *Enumerative combinatorics*. Vol. 2. Cambridge University Press (cité page 266).

Ste05 Steel, Allan (2005). « Conquering inseparability : primary decomposition and multivariate factorization over algebraic function fields of positive characteristic ». In : *Journal of Symbolic Computation*, vol. 40, n°3, p. 1053–1075 (cité page 342).

Ste94 Stern, Jacques (1994). *Fondements mathématiques de l'informatique*. Ediscience international (cité pages 12, 30).

Sto01 Storjohann, Arne (2001). « Deterministic computation of the Frobenius form ». In : *FOCS'01 : IEEE Symposium on Foundations of Computer Science*. Extended abstract. Las Vegas, NV : IEEE Computer Society, p. 368–377 (cité page 181).

Sto02 — (2002). « High-order lifting ». In : *ISSAC'02 : International Symposium on Symbolic and Algebraic Computation*. Éd. par Teo Mora. ACM Press, p. 246–254 (cité page 214).

Sto03 — (2003). « High-order lifting and integrality certification ». In : *Journal of Symbolic Computation*, vol. 36, n°3-4, p. 613–648 (cité page 214).

Sto05 — (2005). « The shifted number system for fast linear algebra on integer matrices ». In : *Journal of Complexity*, vol. 21, n°4, p. 609–650 (cité page 214).

Str69 Strassen, V. (1969). « Gaussian elimination is not optimal ». In : *Numerische Mathematik*, vol. 13, p. 354–356 (cité pages 107, 180).

Str73 — (1972/73). « Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten ». In : *Numerische Mathematik*, vol. 20, p. 238–251 (cité pages 91, 107, 230).

Str77 — (1976/77). « Einige Resultate über Berechnungskomplexität ». In : *Jahresbericht der Deutschen Mathematiker-Vereinigung*, vol. 78, n°1, p. 1–8 (cité page 286).

Str81 — (1981). « The computational complexity of continued fractions ». In : *SYMSAC'81*. Snowbird, Utah, United States : ACM, p. 51–67 (cité page 135).

Str83 — (1983). « The computational complexity of continued fractions ». In : *SIAM Journal on Computing*, vol. 12, n°1, p. 1–27 (cité pages 54, 135).

Str87 — (1987). « Relative bilinear complexity and matrix multiplication ». In : *Journal für die Reine und Angewandte Mathematik*, vol. 375/376, p. 406–443 (cité page 183).

Str90 Strassen, Volker (1990). « Algebraic complexity theory ». In : *Handbook of theoretical computer science, Vol. A*. Amsterdam : Elsevier, p. 633–672 (cité page 180).

SU97 Singer, Michael F. et Felix Ulmer (1997). « Linear differential equations and products of linear forms ». In : *Journal of Pure and Applied Algebra*, vol. 117/118, p. 549–563 (cité page 321).

SV05 Storjohann, Arne et Gilles Villard (2005). « Computing the rank and a small nullspace basis of a polynomial matrix ». In : *ISSAC'05 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 309–316 (cité page 214).

SV55 Svoboda, A. et M. Valach (1955). « Operátorové obvody (Operational circuits) ». In : *Stroje na Zpracování Informací (Information Processing Machines)*, vol. 3, p. 247–295 (cité page 107).

Sýk77 Sýkora, Ondrej (1977). « A fast non-commutative algorithm for matrix multiplication ». In : *Mathematical foundations of computer science (Proc. Sixth Sympos., Tatranská Lomnica, 1977)*. Vol. 53. Lecture Notes in Computer Science. Springer-Verlag, p. 504–512 (cité page 182).

Syl39 Sylvester, James Joseph (1839). « On rational derivation from equations of coexistence, that is to say, a new and extended theory of elimination, *1839–40* ». In : *The Collected mathematical papers of James Joseph Sylvester*. Baker, H. F., p. 40–53 (cité page 135).

Syl40 — (1840). « A method of determining by mere inspection the derivatives from two equations of any degree ». In : *Philosophical Magazine Series 3*, vol. 16, n°101, p. 132–135 (cité page 135).

SZ04 Stehlé, D. et P. Zimmermann (2004). « A binary recursive GCD algorithm ». In : *ANTS-VI*. Vol. 3076. Lecture Notes in Computer Science. Springer-Verlag, p. 411–425 (cité page 136).

SZ94 Salvy, Bruno et Paul Zimmermann (1994). « Gfun : a Maple package for the manipulation of generating and holonomic functions in one variable ». In : *ACM Transactions on Mathematical Software*, vol. 20, n°2, p. 163–177 (cité page 267).

Tak89 Takayama, Nobuki (1989). « Gröbner basis and the problem of contiguous relations ». In : *Japan Journal of Applied Mathematics*, vol. 6, n°1, p. 147–160 (cité page 558).

Tak92 — (1992). « An approach to the zero recognition problem by Buchberger algorithm ». In : *Journal of Symbolic Computation*, vol. 14, n°2-3, p. 265–282 (cité page 576).

Tan74 Tannery, Jules (1874). « Propriétés des intégrales des équations différentielles linéaires à coefficients variables ». Thèse de doctorat ès sciences mathématiques. Faculté des Sciences de Paris. url : http://gallica.bnf.fr (cité page 266).

TD00 Tourigny, Y. et Ph. G. Drazin (2000). « The asymptotic behaviour of algebraic approximants ». In : *The Royal Society of London. Proceedings. Series A. Mathematical, Physical and Engineering Sciences*, vol. 456, n°1997, p. 1117–1137 (cité page 154).

Tel52 Tellegen, B. (1952). « A general network theorem, with applications ». In : *Philips Research Reports*, vol. 7, p. 259–269 (cité page 231).

Tho02 Thomé, E. (2002). « Subquadratic computation of vector generating polynomials and improvement of the block Wiedemann algorithm ». In : *Journal of Symbolic Computation*, vol. 33, n°5, p. 757–775 (cité page 193).

TI61 Takahasi, H. et Y. Ishibashi (1961). « A new method for exact calculation by a digital computer ». In : *Information Processing in Japan*, p. 28–42 (cité page 107).

Too63 Toom, A. L. (1963). « The complexity of a scheme of functional elements simulating the multiplication of integers ». In : *Doklady Akademii Nauk SSSR*, vol. 150, p. 496–498 (cité page 52).

Tra89 Traverso, Carlo (1989). « Gröbner trace algorithms ». In : *Symbolic and Algebraic Computation : International Symposium ISSAC'88 Rome, Italy, July 4–8, 1988 Proceedings*. Éd. par P. Gianni. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 125–138 (cité page 445).

Tre64 Trench, William F. (1964). « An algorithm for the inversion of finite Toeplitz matrices ». In : *Journal of the Society for Industrial and Applied Mathematics*, vol. 12, p. 515–522 (cité page 206).

Tur06 Turner, William J. (2006). « A block Wiedemann rank algorithm ». In : *ISSAC'06 : International Symposium on Symbolic and Algebraic Computation*. New York : ACM Press, p. 332–339 (cité page 193).

Val79 Valiant, L. G. (1979). « The complexity of computing the permanent ». In : *Theoretical Computer Science*, vol. 8, n°2, p. 189–201 (cité page 184).

Van92 Van Loan, Charles (1992). *Computational frameworks for the fast Fourier transform*. Vol. 10. Frontiers in Applied Mathematics. SIAM (cité page 52).

Vas12 Vassilevska Williams, Virginia (2012). « Multiplying matrices faster than Coppersmith–Winograd ». In : *STOC'12 : ACM Symposium on Theory of Computing*. New York, NY : ACM, p. 887–898 (cité page 183).

Vas14 — (2014). « Multiplying matrices in $O(n^{2.373})$ time ». http://theory.stanford.edu/~virgi/matrixmult-f.pdf (cité page 183).

VB91a Vaaler, Jeff et David W. Boyd (1991). « 6613, Mahler's inequality ». In : *The American Mathematical Monthly*, vol. 98, n°5, p. 451–452 (cité page 396).

VB91b Van Barel, Marc et Adhemar Bultheel (1991). « The computation of nonperfect Padé–Hermite approximants ». In : *Numerical Algorithms*, vol. 1, n°3, p. 285–304 (cité page 154).

VB92 Van Barel, M. et A. Bultheel (1992). « A general module-theoretic framework for vector M-Padé and matrix rational interpolation ». In : *Numerical Algorithms*, vol. 3, n°1-4, p. 451–461 (cité page 154).

Vil97 Villard, G. (1997). « Further analysis of Coppersmith's block Wiedemann algorithm for the solution of sparse linear systems ». In : *ISSAC'97 : International Symposium on Symbolic and Algebraic Computation*. Kihei, Maui, Hawaii, United States : ACM Press, p. 32–39 (cité page 193).

Wak70 Waksman, Abraham (1970). « On Winograd's algorithm for inner products ». In : *IEEE Transactions on Computers*, vol. C-19, n°4, p. 360–361 (cité page 180).

Wan78 Wang, Paul S. (1978). « An improved multivariate polynomial factoring algorithm ». In : *Mathematics of Compuptation*, vol. 32, n°144, p. 1215–1231 (cité page 412).

War79 Waring, E. (1779). « Problems concerning interpolations ». In : *Philosophical Transactions of the Royal Society of London*, vol. 59, p. 59–67 (cité page 107).

Wie86 Wiedemann, D. (1986). « Solving sparse linear equations over finite fields ». In : *IEEE Transactions on Information Theory*, vol. IT-32, p. 54–62 (cité pages 193, 231).

Win67 Winograd, S. (1967). « On the number of multiplications required to compute certain functions ». In : *Proceedings of the National Academy of Sciences of the United States of America*, vol. 58, p. 1840–1842 (cité page 180).

Win68 — (1968). « A new algorithm for inner-product ». In : *IEEE Transactions on Computers*, vol. 17, p. 693–694 (cité page 180).

Win71 — (1971). « On multiplication of 2×2 matrices ». In : *Linear Algebra and Applications*, vol. 4, p. 381–388 (cité page 182).

WP03 Wang, Xinmao et Victor Y. Pan (2003). « Acceleration of Euclidean algorithm and rational number reconstruction ». In : *SIAM Journal on Computing*, vol. 32, n°2, p. 548–556 (cité page 153).

WP06 Weimerskirch, André et Christof Paar (2006). « Generalizations of the Karatsuba algorithm for efficient implementations ». Cryptology ePrint Archive. url : https://eprint.iacr.org/2006/224.pdf (cité page 53).

WR75 Wang, Paul S. et Linda Preiss Rothschild (1975). « Factoring multivariate polynomials over the integers ». In : *Mathematics of Computation*, vol. 29, p. 935–950 (cité page 412).

WS79 Welch, L. R. et R. A. Scholtz (1979). « Continued fractions and Berlekamp's algorithm ». In : *IEEE Transactions on Information Theory*, vol. 25, n°1, p. 19–27 (cité page 153).

Yam00 Yamamoto, Tetsuro (2000). « Historical developments in convergence analysis for Newton's and Newton-like methods ». In : *Journal of Computational and Applied Mathematics*, vol. 124, n°1-2, p. 1–23 (cité page 76).

Yap00 Yap, Chee (2000). *Fundamental problems in algorithmic algebra*. Oxford University Press (cité pages 13, 136).

Yap11 — (2011). « A real elementary approach to the master recurrence and generalizations ». In : *Theory and applications of models of computation*. Vol. 6648. Lecture Notes in Computer Science. Tokyo, Japan : Springer-Verlag, p. 14–26 (cité page 30).

Ypm95 Ypma, Tjalling J. (1995). « Historical development of the Newton–Raphson method ». In : *SIAM Review*, vol. 37, n°4, p. 531–551 (cité page 76).

Yun76 Yun, David Y. Y. (1976). « On square-free decomposition algorithms ». In : *SYMSAC'76 : ACM Symposium on Symbolic and Algebraic Computation*. Yorktown Heights, New York, United States : ACM, p. 26–35 (cité pages 136, 342).

Yun77 — (1977). « On the equivalence of polynomial GCD and squarefree factorization problems ». In : *MACSYMA Users' Conference*. NASA, Langley Res. Center, Washington D.C., United States, p. 65–70 (cité page 136).

Zas69 Zassenhaus, Hans (1969). « On Hensel factorization I ». In : *Journal of Number Theory*, vol. 1, n°1, p. 291–311 (cité pages 356, 396).

Zei90 Zeilberger, Doron (1990). « A holonomic systems approach to special functions identities ». In : *Journal of Computational and Applied Mathematics*, vol. 32, n°3, p. 321–368 (cité page 576).

Zie68 Zierler, N. (1968). « Linear recurring sequences and error-correcting codes ». In : *Error Correcting Codes (Proc. Sympos. Math. Res. Center, Madison, Wis., 1968)*. Wiley, p. 47–59 (cité page 153).

Zip79 Zippel, Richard (1979). « Probabilistic algorithms for sparse polynomials ». In : *Symbolic and Algebraic Computation. Eurosam '79, An International Symposium on Symbolic and Algebraic Manipulation, Marseille, France, June 1979*. Vol. 72. Lecture Notes in Computer Science. Springer-Verlag, p. 216–226 (cité page 193).

Zip90 Zippel, R. (1990). « Interpolating polynomials from their values ». In : *Journal of Symbolic Computation*, vol. 9, n°3, p. 375–403 (cité page 231).

Zip93 Zippel, Richard (1993). *Effective polynomial computation*. Kluwer Academic Publishers (cité pages 13, 413).

ZLS15 Zhou, Wei, George Labahn et Arne Storjohann (2015). « A deterministic algorithm for inverting a polynomial matrix ». In : *Journal of Complexity*, vol. 31, n°2, p. 162–173 (cité page 214).

Źra10 Źrałek, Bartosz (2010). « Using partial smoothness of $p-1$ for factoring polynomials modulo p ». In : *Mathematics of Computation*, vol. 79, p. 2353–2359 (cité page 356).

Liste des notations

$\mathbb{N}$ est l'ensemble des entiers naturels.

$\mathbb{Z}$ est l'ensemble des entiers relatifs.

$\mathbb{Q}$ est l'ensemble des nombres rationels.

$\mathbb{R}$ est l'ensemble des nombres réels.

$\mathbb{C}$ est l'ensemble des nombres complexes.

$\mathrm{Re}\, z$ est la partie réelle du nombre complexe z.

M est une fonction de coût pour le produit des polynômes à une variable, voir page 50.

$\mathsf{M}_{\mathbb{Z}}$ est une fonction de coût pour le produit des entiers, voir page 51.

$\mathsf{MM}(n)$ est une fonction de coût pour le produit des matrices $n \times n$, voir page 163.

$\mathsf{MM}(n,d)$ est une fonction de coût pour le produit des matrices $n \times n$ à coefficients des polynômes de degrés au plus d, voir page 167.

$\lfloor x \rfloor$ est le plus grand entier inférieur ou égal au réel x.

$\lceil x \rceil$ est le plus petit entier supérieur ou égal au réel x.

$\lceil x \rfloor$ est l'entier le plus proche du réel x (par défaut, l'entier immédiatement inférieur si x est un demi-entier).

$\|(v_1,\ldots,v_n)\|_1 = |v_1| + \cdots + |v_n|$ représente la norme 1 de v.

$\|(v_1,\ldots,v_n)\|_2 = (|v_1|^2 + \cdots + |v_n|^2)^{1/2}$ représente la norme 2 de v.

$\|(v_1,\ldots,v_n)\|_\infty = \max(|v_1|,\ldots,|v_n|)$ représente la norme infinie de v.

$(u \mid v)$ représente le produit scalaire entre deux vecteurs u et v ; pour des vecteurs donnés en coordonnées, $u = (u_1, \ldots, u_n)$ et $v = (v_1, \ldots, v_n)$, il s'agit de la somme $u_1 v_1 + \cdots + u_n v_n$.

$f(n) = \mathrm{O}(g(n))$ signifie qu'il existe une constante C telle que $|f(n)| \leq \mathrm{C}|g(n)|$ lorsque n est suffisamment grand.

$f(n) = \tilde{\mathrm{O}}(g(n))$ signifie qu'il existe un entier k tel que $f(n) = \mathrm{O}(g(n)\log^k(\max(|g(n)|, 2)))$.

$\mathrm{S}(\mathrm{X}) = \mathrm{O}(\mathrm{X}^n)$ signifie que S est une série de valuation au moins n.

card(E) est le cardinal de l'ensemble E.

disc(F) est le discriminant du polynôme F, voir page 118.

ppcm(F, G) est le plus petit commun multiple de F et G.

pgcd(F, G) est le plus grand commun diviseur de F et G.

rem(F, G) est le reste de la division euclidienne de F par G.

Res(F, G) est le résultant des polynômes F et G, voir page 117.

Syl(F, G) est la matrice de Sylvester des polynômes F et G, voir page 116.

psc(F) est la partie sans carré de F.

fsc(F) est la factorisation sans carré de F.

fsep(F) est la factorisation séparable de F.

firr(F) est la factorisation irréductible de F.

val(S) est la valuation de la série S.

$\mathbf{V}_k(\mathrm{I})$ est la variété définie par l'idéal I sur k.

$\mathbf{I}(\mathrm{E})$ est l'idéal des polynômes s'annulant sur l'ensemble E.

$\exp_<(f)$ est l'exposant privilégié (de tête) du polynôme f pour l'ordre $<$ sur les monômes.

$\mathrm{mt}_<(f)$ est le monôme de tête du polynôme f pour l'ordre $<$ sur les monômes.

$\mathrm{ct}_<(f)$ est le coefficient de tête du polynôme f pour l'ordre $<$ sur les monômes.

$\mathrm{tt}_<(f)$ est le terme de tête du polynôme f pour l'ordre $<$ sur les monômes.

`lex` est l'ordre lexicographique, voir page 426.

`grlex` est l'ordre lexicographique gradué, voir page 426.

`tdeg` est l'ordre lexicographique renversé gradué, voir page 427.

${}_p\mathrm{F}_q\left(\begin{matrix}a_1, \ldots, a_p \\ b_1, \ldots, b_q\end{matrix} \middle| x\right)$ est la série hypergéométrique généralisée.

Liste des figures et algorithmes

Index des auteurs

L

M

N

Z

Index

D

E

F

G

H

I

N

O

P

T

U