Algorithmes pour l'étude des solutions réelles des systèmes polynomiaux

Fabrice Rouillier

UNIVERSITE
PIERRE & MARIE CURIE
LA SCIENCE A PARIS

Mémoire d'habilitation à diriger des recherches

(Spécialité informatique)

Algorithmes pour l'étude des solutions réelles des systèmes polynomiaux

Fabrice Rouillier

Rapporteurs :

- ***H. Hong*** – Professeur – North Carolina State University – Raleigh - USA
- ***T. Recio*** – Professeur – Universitad de Cantabria – Santander - Espagne
- ***B. Sturmfels*** – Professeur – University of California – Berkeley – USA

Présenté le 29 Mars 2007 au Laboratoire d'Informatique de Paris VI

Jury :

- ***J.-D. Boissonnat*** – Directeur de Recherche INRIA – INRIA Sophia-Antipolis – France
- ***J.-M. Chesneaux*** – Professeur - Université Pierre et Marie Curie – Paris – France
- ***J.-C. Faugère*** – Chargé de Recherche CNRS – Laboratoire d'Informatique de Paris VI – France
- ***D. Lazard*** – Professeur émérite – Université Pierre et Marie Curie – Paris – France
- ***T. Recio*** – Professeur – Universitad de Cantabria – Santander – Espagne
- ***M.-F. Roy*** – Professeur – Université de Rennes I – Rennes - France

Remerciements

En imprimant la couverture de ce document, je me suis dit que j'avais beaucoup de chance d'avoir pu réunir de telles personnalités autour de cette habilitation.

Je suis très fier que H. Hong, T. Recio et B. Sturmfels aient immédiatement accepté d'évaluer par écrit mon travail et heureux que ceux qui ont encadré mon parcours depuis le début (M.-F. Roy et D. Lazard) soient présents dans le jury.

Je remercie chaleureusement J.-D. Boissonnat et J.-M. Chesneaux d'avoir accepté d'être présents à ma soutenance, représentant à la fois l'ouverture thématique qui m'est chère, mais également les instituts qui m'ont accueillis depuis mes débuts en tant que chercheur permanent.

La souplesse des règles de l'UPMC m'a permis d'inviter mon collègue « de toujours » J.-C. Faugère à participer au jury. Le travail présenté dans ce document lui doit beaucoup, mon parcours peut-être plus encore, et j'espère que nous continuerons à partager, dans les années qui viennent, notre goût commun pour le ….. risque.

En écrivant le mot « risque », il m'est difficile d'oublier dix années d'un parcours riche en obstacles de toutes sortes et j'aimerais ici remercier tous ceux qui m'ont aidé dans les pires moments. Je pense bien sûr à mon ami P. Zimmermann et, plus généralement, aux anciens du projet PolKA du LORIA, ainsi qu'au soutien sans faille des membres des projets SPACES puis SALSA (Mohab, les Philippe(s),..) sans oublier ceux de l'équipe SPIRAL (Gwénael, Sébastien, et j'en oublie certainement …).

Fabrice.

« On fait la science avec des faits, comme on fait une maison avec des pierres : mais une accumulation de faits n'est pas plus une science qu'un tas de pierres n'est une maison »

Henri Poincaré

« Ce qui est affirmé sans preuve peut être nié sans preuve »

Euclide

« Ceux qui aiment marcher en rangs sur une musique : ce ne peut être que par erreur qu'ils ont reçu un cerveau, une moelle épinière leur suffirait amplement »

Albert Einstein

À Chloé, Julien et Véronique,

« Les passions sont toutes bonnes de leur nature et nous n'avons rien à éviter que leurs mauvais usages ou leurs excès »

René Descartes

« Douter de tout ou tout croire sont deux solutions également commodes, qui l'une et l'autre nous dispensent de réfléchir »

Henri Poincaré

« N'entretiens pas l'espoir de ce qui ne peut être espéré »

Pythagore

Résumé

Cette présentation résume un ensemble de méthodes générales pour la résolution des systèmes d'équations (et d'inéquations/inégalités) polynomiales, axées principalement sur l'étude de leurs racines réelles ; Il s'agit essentiellement de « résoudre » des systèmes admettant ou non une infinité de solutions, dépendant ou non de paramètres. Le sujet étant vaste, on le contraint en imposant que les algorithmes soient exacts ou certifiés, ou, en d'autres termes, que les résultats ne présentent aucune ambigüité du point de vue de l'utilisateur (nombre ou structure des solutions, approximations numériques lorsque cela a un sens, caractère réel des solutions, etc.).

Le travail exposé a été effectué avec deux objectifs principaux : progresser dans la mise au point de méthodes effectives utiles en géométrie réelle, mais surtout proposer quelques alternatives crédibles aux outils standards de calcul scientifique, avec pour objectif de répondre à quelques problèmes ouverts (ou réputés difficiles) dans divers domaines applicatifs.

Ce bilan de plusieurs années de travail est, en particulier, l'occasion de montrer les efforts spécifiques a priori nécessaires pour arriver à quelques résultats probants (implantation efficace, coût de la certification, balance entre les efficacités théoriques et pratiques) et s'articule autour de quelques applications « fil rouge » ayant motivé les principaux choix théoriques, algorithmiques voir techniques.

Abstract

This presentation summarizes a set of general methods for solving systems of polynomial equations (with or without inequalities), mainly centered on the study of their real roots; The main subject is the "resolution" of systems with an arbitrary number of solutions (finite or infinite), depending or not on parameters. Since the field is vast, one constrains it by imposing it that the algorithms are exact or certified or, in other words, that the results are never ambiguous from the user's point of view (number or structure of the solutions, numerical approximations when it makes sense, real character of the solutions, etc.).

The exposed work was carried out with two principal objectives: to progress in the development of useful effective methods in real geometry, but especially to propose some credible alternatives to the standard tools for scientific computations, with the objective of answering to some open problems (or considered difficult) in various applications.

This assessment of several years of work is, in particular, the occasion to show the specific efforts which seem to be necessary to obtain convincing results (efficient implementations, cost of the certification/exactness, balance between theoretical and practical efficiency, etc.) and is articulated around some selected applications which justify the main theoretical, algorithmic but also technical choices.

Table des matières

Table des matières 1

1 Introduction 3

1.1 Quelques remarques et objectifs généraux 3
1.2 La feuille de route 7
1.3 Un guide de lecture 8

2 Polynômes en une variable 11

2.1 Introduction 11
2.2 Généralités sur les méthodes à base de bissection 13
2.2.1 Utilisation de suites de Sturm 13
2.2.2 Utilisation de la règle des signes de Descartes 14
2.3 Une description unifiée des algorithmes de bissection 15
2.4 Un algorithme de Descartes efficace et optimal en mémoire 17
2.5 Le cas de polynômes exprimés dans la base de Bernstein 19
2.6 Complexité - Choix de la variante la plus adaptée 21
2.7 Utilisation d'arithmétique en précision fixe 23
2.8 Expériences 24
2.9 Extensions 27
2.10 Conclusion 28

3 Systèmes zéro-dimensionnels 31

3.1 Introduction 31
3.2 Des systèmes zéro-dimensionnels à l'algèbre linéaire 31
3.3 Résolution Symbolique : La Représentation Univariée Rationnelle (RUR) 34
3.4 Algorithmes et complexité 36
3.4.1 Calcul d'un candidat RUR - algorithme Classic 36
3.4.2 Complexité binaire de l'algorithme Classic 38
3.4.3 Calcul d'un candidat RUR - l'algorithme BS/GS et sa complexité 39
3.4.4 Choix d'une stratégie pour le calcul d'un candidat-RUR 40
3.4.5 RUR et candidat-RUR 41
3.4.5.1 Calculer un élément séparant sans connaître de candidat-RUR 41
3.4.5.2 Vérifier si un élément est séparant connaissant un candidat-RUR 42
3.5 Un algorithme efficace pour le calcul de RUR 44
3.5.1 Utilisation de calcul modulaire 45
3.5.2 Le cas des idéaux *shape position* 46
3.5.3 L'algorithme général 49
3.6 Extension aux systèmes avec inégalités/inéquations 49
3.7 Résolution certifiée de systèmes zéro-dimensionnels 50
3.8 Conclusion 52

4 Systèmes de dimension positive 55

4.1 Introduction 55
4.1.1 Le cas des variétés définies par une unique équation. 57
4.1.2 Le cas des variétés définies par un système d'équations. 60
4.1.3 Expériences et conclusions 63

5 Systèmes dépendant de paramètres 65

5.1 Introduction . . . 65
5.2 Les variétés discriminantes dans le cas complexe . . . 67
5.3 Les variétés discriminantes dans le cas réel . . . 69
5.4 Variétés discriminantes, résolution de systèmes et effectivité . . . 70
5.4.1 Calcul effectif de variétés discriminantes . . . 70
5.4.2 Utilisation de variétés discriminantes pour la résolution de systèmes paramétrés . 71
5.5 Le cas des systèmes « bien formulés » . . . 72
5.6 Un algorithme général pour le calcul de variété discriminante . . . 74
5.6.1 Les algorithmes externes . . . 75
5.6.2 Calculs inconditionnels . . . 76
5.6.3 Le calcul des points critiques et singuliers . . . 78
5.6.4 Le cas des systèmes les plus courants . . . 79
5.6.5 L'algorithme général dans le cas complexe . . . 79
5.6.6 L'algorithme général dans le cas réel . . . 81
5.6.7 Une CAD optimisée pour l'utilisation des variétés discriminantes . . . 81
5.7 Conclusion et perspectives . . . 83

6 Quelques applications . . . 85

6.1 Systèmes zéro-dimensionnels . . . 85
6.1.1 Le problème géométrique direct des robots parallèles . . . 85
6.1.2 Calcul de la topologie de ridges . . . 92
6.1.2.1 La structure algébrique implicite des ridges . . . 94
6.1.2.2 Un algorithme de tracé certifié . . . 96
6.1.2.3 Calcul de la topologie des ridges . . . 98
6.2 Systèmes de dimension positive . . . 100
6.2.1 Le problème d'interpolation de Birkhoff . . . 100
6.2.2 Synthèse de bancs de filtres de dimension 2 non séparables . . . 102
6.2.2.1 Le problème . . . 105
6.2.2.2 Application sur un cas connu . . . 108
6.2.2.3 Synthèse de nouveaux filtres . . . 109
6.3 Systèmes dépendant de paramètres . . . 111
6.3.1 Classification d'une famille de robots série . . . 111
6.3.1.1 Résolution ad-hoc . . . 113
6.3.1.2 Résolution par les variétés discriminantes . . . 116
6.3.2 Certification d'une méthode d'optimisation en traitement du signal . . . 116

7 Conclusion et perspectives . . . 121

Bibliographie . . . 127

Chapitre 1
Introduction

1.1 Quelques remarques et objectifs généraux

Les travaux présentés dans ce document portent sur la mise au point, l'implantation et l'application d'algorithmes certifiés et efficaces pour la géométrie algébrique réelle. Le domaine étant vaste et pouvant être abordé de multiples façons, cette première section a pour but de donner un cadre précis et de fixer les objectifs poursuivis.

Sur la notion d'algorithme

Par algorithme certifié, il faudra comprendre « méthode infaillible », c'est à dire calculant en un nombre fini d'étapes, quelque soit l'entrée, un résultat sans ambiguïté du point de vue de l'utilisateur. Par exemple, un réel ne doit en aucun cas être assimilé à un complexe de petite partie imaginaire ou encore un point multiple ne doit pas être confondu avec un ensemble de points très proches. Sont en particulier exclus les algorithmes pouvant structurellement entrer dans une « boucle infinie » ou encore retournant un résultat faux dans certains cas, aussi peu nombreux soient-ils. On autorise cependant une réponse de type « je ne sais pas faire ».

Ceci n'empêche en rien d'avoir des algorithmes dédiés à certaines classes de sous-problèmes dès lors que les hypothèses induites sur l'entrée sont vérifiables par au moins un autre algorithme. Par contre, on bannit les méthodes dépendant d'un choix « générique », le plus souvent aléatoire en pratique, rendant l'algorithme probabiliste, même si la probabilité d'erreur est faible, à moins qu'un autre algorithme puisse être utilisé, éventuellement a posteriori, pour vérifier la « généricité » du choix effectué.

Ce type de contraintes permet d'avoir des « boites noires » solides, bien spécifiées, et remplaçables à volonté puisqu'elles calculent ou prennent en entrée des objets mathématiques clairement définis en dehors de tout algorithme. Les solutions présentées dans ce document s'articulent de façon gigogne et l'esprit « boite noire » permet alors de considérer chaque composant de façon indépendante, ceux-ci répondant le plus souvent à une question intermédiaire d'intérêt.

Ces spécifications fortes permettent également de se positionner clairement vis à vis des autres méthodes. L'exactitude et la fiabilité à tout prix incitent particulièrement à traiter des cas pathologiques, de préférence inatteignables par d'autres algorithmes.

Sur la notion d'efficacité

Les indicateurs d'« efficacité » sont, bien sûr, très relatifs, les outils de mesure étant le plus souvent la complexité théorique asymptotique ou simplement le chronomètre. L'objectif final étant l'implantation avec comme point de mire de résoudre de nouvelles classes de problèmes, beaucoup de précautions doivent être prises.

En premier lieu, les notations $O(.)$ seront à considérer avec prudence. En effet, borne de Bézout oblige, on ne peut que s'attendre à des algorithmes exponentiels en certains paramètres des entrées (par exemple nombre de variables) ou alors polynomiaux mais en des entrées de tailles exponentielles en les paramètres du problème initial. Ainsi, chaque constante devient excessivement importante puisque, par exemple, une routine exponentielle peut être lancée un nombre exponentiel de fois sans que la complexité globale en $O(.)$ ne s'en trouve changée, ce qui, d'un point de vue pratique, est bien sûr une aberration.

D'autre part, la différence entre la complexité arithmétique (nombre d'opérations arithmétiques chaque opération étant supposée de coût unitaire quelles que soient les tailles des opérandes) et la complexité binaire (nombre d'opérations binaires) induit parfois de grossières erreurs d'interprétation quant à l' «efficacité» d'une méthode : selon le modèle, on peut éventuellement inverser la conclusion.

Enfin, il faut se rappeler qu'il ne s'agit, en général, que de bornes supérieures dans le pire des cas, qui ne mesurent donc pas forcément le comportement moyen des méthodes et ne permettent que très rarement les comparaisons objectives sur leur rapidité effective. Par exemple, les bases de Gröbner sont de complexité doublement exponentielle en le nombre de variables dans le pire des cas, mais simplement exponentielles dans beaucoup de situations, voir dans la plupart des situations pratiques.

L'ensemble des résultats présentés dans ce document s'appuie sur quelques paris et particulièrement celui que l'on saurait un jour « calculer » efficacement un idéal. Ce type de stratégie n'est pas isolé puisque, par exemple, les algorithmes de [13] utilisent massivement les bases de Gröbner, en évitant toutefois de les calculer à l'aveugle lorsque cela est possible. Ici, comme par exemple dans [135], on considère que l'on dispose, par défaut, d'une boite noire permettant de calculer ce type d'objet. Il faut alors souligner que les algorithmes présentés ne sont donc devenus « efficaces » qu'avec l'arrivée d'une nouvelle génération de méthodes [44],[50], expliquant partiellement que leurs études de complexité aient été effectuées, le plus souvent, à part et longtemps après leur mise au point.

Un bon indicateur est certainement les classes de problèmes atteignables par les méthodes considérées et en particulier les applications pour lesquelles les algorithmes proposés sont une valeur ajoutée par rapport à l'état de l'art.

Un autre indicateur important d'« efficacité » est également l'occupation mémoire induite par un algorithme. Il n'est pas rare, en effet, de voir des méthodes efficaces en termes temps de calcul ou de complexité arithmétique voir binaire, mais sans conséquences pratiques à cause d'une occupation mémoire prohibitive.

La recherche d' « efficacité » est donc a priori une balance entre le théoriquement rapide et le praticable. Un objectif intellectuellement satisfaisant est alors de mettre au point des algorithmes appartenant au moins aux bonnes classes de complexité théorique mais avant tout praticables d'un point de vue implantation et utilisation à des fins applicatives.

Sur les objectifs algorithmiques

Le sujet général abordé est celui de la géométrie réelle effective. On peut classer en deux catégories les recherches effectuées dans ce domaine.

- les développements pour des problèmes spécifiques (étude de courbes, surfaces, etc.) ou soit le nombre de variables est fixé, soit la classe de problèmes est restreinte (polynômes en une variable, systèmes admettant un nombre fini de solutions, etc.).

- les développements menant à la résolution de problèmes généraux, en particulier l'élimination des quantificateurs.

Les problèmes généraux d'élimination de quantificateurs se ramènent naturellement à l'étude des ensembles semi-algébriques basiques (Tarski). L'étude de cette dernière classe d'objets est donc essentielle et on s'y restreindra dans ce document. L'ambition est de se situer dans la deuxième catégorie ou il existe essentiellement deux lignes de pensée :

- le schéma récursif sur les variables où l'outil de base est la résolution des polynômes en une variable (résultants, suites de Sturm, etc); Pour résumer, l'étude des ensembles semi-algébriques se fait par projections successives, oubliant à chaque étape une coordonnée.

- le schéma mathématiquement gigogne multipliant les outils de base plus complexes comme la résolution des systèmes zéro-dimensionnels, l'étude des hypersurfaces, celle des variétés algébriques, semi-algébriques, etc.

La première approche est informatiquement pragmatique, l'utilisation d'objets simples favorisant l'implantation des algorithmes. L'essentiel des stratégies existantes est résumé par le calcul d'une Décomposition Cylindrique Algébrique [30], c'est à dire la décomposition de l'espace ambiant en cellules (homéomorphes à un pavé de $\mathbb{R}^k$, $k = 0...n$) dans lesquelles une séquence de polynômes ont des signes constants. Bien que de nombreux raffinements aient été proposés, il n'en demeure pas moins que l'évolution de ce type de stratégie est limitée, la complexité processus récursif étant intrinsèquement doublement exponentielle en le nombre de variables, ce phénomène étant « visible » en pratique.

La seconde approche, dont une instance complète peut-être trouvée dans [13], multiplie les objets intermédiaires. Pour prendre l'exemple de l'environnement décrit dans [13], par déformations successives, l'étude d'un semi-algébrique est ramenée à celle d'un ensemble algébrique puis à celle d'une hypersurface compacte et lisse et enfin à la résolution de systèmes zéro-dimensionnels par l'étude des points critiques d'une fonction bien choisie. L'étude théorique proposée [13], montre l'existence d'un algorithme de bonne complexité asymptotique arithmétique (optimale ou quasi optimale), calculant un résultat de bonne taille (simplement exponentiel) dans le pire cas, pour chaque problème intermédiaire.

Si l'on prétend résoudre certains problèmes intermédiaires (systèmes admettant un nombre fini de solutions, test existentiels pour les variétés algébriques ou semi-algébriques, etc.), importants du point de vue des applications, une stratégie raisonnable suivra la seconde approche qui ne se trouve pas intrinsèquement bloquée par une complexité prohibitive.

En pratique, les méthodes « naturelles » sous-jacentes s'avèrent toutefois excessivement difficiles, voir impossibles, à mettre en oeuvre. Même si elle est bien étudiée, la complexité asymptotique ne donne pas, dans ce cas, un reflet réaliste du comportement pratique de ces algorithmes, à cause de structures de données manipulées (par exemple utilisation d'une arithmétique dans le corps des séries de Puiseux) mais surtout de certains artifices utilisés pour maintenir des bornes acceptables (par exemple prendre la somme des carrés des équations définissant une variété algébrique réelle simplifie mathématiquement le problème mais augmente artificiellement le degré des polynômes utilisés dans les calculs). La nature intrinsèquement exponentielle des objets/algorithmes permet bon nombre d'écarts en théorie, beaucoup moins en pratique.

Sur le travail applicatif

Avec les objectifs d'exactitude et de certification des résultats évoqués plus haut, les méthodes de calcul employées ou mises au point dans ce document s'inscrivent dans le domaine du calcul formel. La matière étant récente et notoirement pointée du doigt pour l'inefficacité des solutions qu'elle propose, les applications jouent alors un rôle essentiel dans le travail qui est présenté. Elles permettent, en effet, de valider l'approche sur des problèmes issus d'autres domaines et réputés difficiles.

Dans le cas zéro-dimensionnel, si un système algébrique ne présente pas de structure particulière, le nombre de ses solutions est simplement exponentiel en le nombre de variables non linéaires (le traitement des systèmes linéaires ne faisant pas partie des thèmes abordés dans ce document). L'utilisation à l'aveugle d'algorithmes généraux sur des systèmes admettant de nombreuses variables est donc toujours totalement inadaptée, que ce soit avec des méthodes numériques ou avec des méthodes algébriques.

Il est usuel, en calcul numérique, d'utiliser la structure des équations modélisant le problème (totalement ou en partie) pour en modifier le conditionnement et/ou l'étudier localement. Le caractère exponentiel du nombre de solutions rend nécessaire un travail analogue, dans le cadre de l'utilisation de méthodes formelles, sur les équations ainsi que sur la modélisation des problèmes posés. La notion de « conditionnement » n'est simplement pas la même : par exemple, on évitera en général les linéarisations et on calculera les singularités plutôt que de les contourner, on préférera les tests à zéro aux inégalités, etc.

Ce travail systématique sur la modélisation et la mise en équations, courant chez les numériciens, est peu répandu en calcul formel. Dans le travail présenté dans ce document, quelques applications ont servi de fil conducteur pour affiner progressivement et simultanément la spécification des algorithmes de calcul, la définition des objets mathématiques qui leur sont utiles et le travail d'implantation nécessaire à leur validation en pratique. Étudier en même temps ces différents aspects a permis de dériver, de résolutions initialement ad hoc, un certain nombre d'algorithmes généraux permettant de résoudre efficacement certaines classes de problèmes inatteignables que ce soit par des méthodes numériques ou algébriques.

Une conséquence est que les calculs ne sont effectués que très rarement sur des problèmes modélisés par des méthodes « standard » (numériques), les systèmes d'équations résultants étant généralement inadaptés aux méthodes « exactes ». En effet, les modélisations favorisant les méthodes « standard » vident souvent le problème d'un contenu utile par des opérations de reconditionnement (recherche de systèmes convexes, linéarisations, etc.). Sur les exemples traités, la reformulation, ainsi qu'un pré-traitement symbolique, ont permit quasi systématiquement d'obtenir des modélisations présentant significativement moins de variables qu'avec des modélisations classiques mais surtout beaucoup plus exploitables par des méthodes formelles.

Sur le travail d'implantation

Au regard des objectifs proposés, l'implantation des algorithmes est un maillon essentiel et c'est peut-être une des tâches les plus difficiles à appréhender, faute de bibliographie. En effet, une spécificité du calcul formel est de manipuler des objets aussi multiples que complexes d'un point de vue informatique.

Les difficultés majeures résident dans la variation de la taille des objets au cours du temps et dans la mauvaise maîtrise du comportement des algorithmes (nombre d'opérations et occupation mémoire). On rejoint ici l'utilité toute relative des études de complexité théorique ne donnant qu'une information partielle sur le comportement réel des algorithmes : selon l'implantation, le choix des outils de base (par exemple arithmétique ou encore algèbre linéaire) une même méthode peut passer de l'instantané à l'impossible sur un même problème.

Une mauvaise compréhension de l'aspect informatique peut ainsi engendrer de mauvais choix stratégiques. En ayant à l'esprit que la notion d'efficacité englobe le temps de calcul mais également les aspects d'utilisation de la mémoire (un algorithme de bonne complexité peut être impraticable pour cause d'occupation mémoire abusive), ce texte sera parsemé de « ruses » algorithmiques ayant pour objectif que de favoriser une implantation efficace sans pour autant nécessairement apporter de progrès « théorique » au sens classique du terme.

Par contre, ce document ne donnera que peu de détails sur certains aspects plus techniques comme la gestion de mémoire, la localisation des données ou encore le choix des langages et des stratégies de programmation, par souci d'homogénéité, mais aussi simplement parce que ces travaux ne sont pas publiés à ce jour faute d'espace pour le faire.

Sur le positionnement en calcul scientifique

Une question redondante est l'utilité et, plus généralement, le « passage à l'échelle » de méthodes exactes ou certifiées en calcul scientifique. Les attentes sont évidemment très fortes puisque le travail présenté pourrait apparaître, dans ses objectifs, comme une sorte de remède miracle à certains grands maux du calcul scientifique. L'erreur serait de penser que l'on peut produire des substituts pouvant être intégrés à des schémas classiques (numériques) en l'état. Au contraire, les méthodes proposées s'accompagnent nécessairement d'un transfert de savoir faire, remettant en cause beaucoup de choses en amont (mises en équations, modélisations etc.) comme en aval (outils de développement informatique, implantations, etc.), le tout rendant difficile l'évaluation de l'apport intrinsèque et surtout impossible l'évaluation de l'apport de projets à court terme.

Cette dichotomie est flagrante au regard des moyens usuels de diffusion de l'information (conférences et journaux) : les liens entre ces communautés est faible en général. Pour utiliser un raccourci trivial, les techniques d'analyse numérique sont intrinsèquement basées sur le « tant que tout va bien on continue » sous entendu cherchons une solution particulière acceptable puis élargissons le domaine. Le calcul exact fonctionne généralement à l'inverse « en dehors de ce qui va mal, tout va bien », sous entendu « analysons les dégénérescences et déduisons ensuite les domaines ou tout se passe bien ». Les raisons sont structurelles : l'outil numérique est incapable de gérer en toute rigueur l'égalité (calcul flottant), alors que c'est au contraire la spécificité du calcul exact.

Le calcul exact peut être utilisé à différents niveaux : en amont de la résolution pour simplifier le problème (réécriture des équations, voir par exemple le travail sur les bancs de filtres et son application à la compression d'images), pour la résolution complète d'une formulation du problème (certaines méthodes s'avèrent maintenant tout à fait concurrentielles par exemple la méthode de Descartes pour les polynômes en une variable) ou simplement pour certifier la méthode numérique qui sera employée pour le résoudre (voir application « réseaux sans fils »).

La difficulté est de savoir ou placer le curseur en fonction du résultat désiré (niveau de certification) et des contraintes de l'application visée (temps de calcul). Le travail sur les « ridges » en géométrie algorithmique présenté dans ce document illustre ces différents niveaux d'utilisation.

Comme évoqué plus haut, l'expérimentation et le travail applicatif sont d'une extrême importance pour opérer les réglages adéquats et tirer profit de l'état de l'art. Une erreur serait de laisser croire que le curseur pourrait être positionné complètement du coté calcul certifié. Par contre, une remarque essentielle est que quelle que soit son utilisation, le calcul symbolique, même si il n'intervient que pour une part infime d'un processus complet, peut être le chaînon manquant permettant simplement à un algorithme d'aboutir (voir application à la planification de trajectoire des robots parallèles). On ne parle donc plus de temps de calcul mais déjà d'avoir une méthode qui fonctionne et répond à la question.

1.2 La feuille de route

Si l'on considère la méthodologie choisie, et en particulier le guidage des travaux par des applications « fil rouge », un découpage naturel des problèmes à résoudre peut être effectué :

- **Les systèmes zéro-dimensionnels** (admettant un nombre fini de solutions complexes - ce qui inclut la résolution des polynômes en une variable). Pour ce type de problèmes, les questions posées sont bien identifiées (nombre de points complexes/réels, isolation des racines, approximation numérique, calcul des points multiples et multiplicités, signes d'expressions algébriques en les zéros d'un système, etc.) et les réponses faciles à spécifier.

- **Les systèmes paramétrés** (admettant un nombre fini de solutions pour presque toutes les valeurs consistantes des paramètres). Ils représentent une part importante des problèmes de calcul naturellement soumis. Les questions sont bien identifiées (nombre de solutions en fonction des paramètres, signes d'expressions algébriques en fonction des valeurs des paramètres, paramétrisations rationnelles, description des solutions, topologie des solutions etc.) mais les réponses à apporter sont difficiles à spécifier en général (par exemple, comment décrire une région de l'espace des paramètres).

- **Les systèmes généraux de dimension positive** (admettant une infinité de zéros complexes). Pour ce qui est des applications, les questions possibles sont relativement mal identifiées. Bien que certains problèmes se résument à déterminer si un système général admet ou non des solutions réelles et éventuellement en calculer un nombre fini, la ques-

tion la plus fréquente reste vague : « pouvez-vous résoudre le système ?». Ce type de question n'a aucun sens dans l'absolu puisqu'il est alors impossible de spécifier la sortie d'un algorithme y répondant. Un problème central pour les ensembles semi-algébriques généraux est le calcul d'un point par composante connexe; d'une part parce que cela répond au problème de décider si un système admet ou non des solutions, mais surtout parce que ce type de calcul sert de « boite noire » pour d'autres algorithmes comme la résolution des systèmes paramétrés présentée dans ce document ou alors certaines méthodes d'élimination de quantificateurs (voir [13]).

- **Les problèmes généraux d'élimination de quantificateurs**. Ils ne seront pas abordés dans le corps de ce document. Bien que certains développements présentés y soient directement reliés (systèmes paramétrés - calcul d'un point par composante connexe d'un semi-algébrique), ils n'augmentent l'état de l'art sur des problèmes généraux comme l'élimination des quantificateurs que par ricochet et ne constituent pas un progrès fondamental sur les stratégies existantes.

Cette présentation simplifiée des questions que ce document aborde occulte certains aspects plus théoriques comme les calculs de topologie en général, les problèmes de cartes routières, etc. Il ne s'agit pas de négliger l'importance de ces questions mais force est de constater que pour le moment leur impact applicatif est faible.

1.3 Un guide de lecture

Le plan de ce document est classique, présentant les applications dans le dernier chapitre. Ce choix a été effectué pour en faciliter une lecture séquentielle mais ne reflète pas forcément la part essentielle de l'activité applicative dans la mise au point des méthodes et algorithmes proposés. Le chapitre 6 doit donc être lu en parallèle avec les autres. Les applications présentées ont été sélectionnées pour permettre de mesurer les progrès effectués mais également pour tenter de donner une image précise du cheminement qui les a engendrés.

Les applications « fil-rouge »

Chaque résultat décrit dans ce document est adossé à au moins une application « fil rouge », inatteignable avant avant l'étude :

- *Systèmes zéro-dimensionnels (incluant les polynômes en une variable).* La résolution du problème géométrique direct des robots parallèles a été pendant longtemps un challenge pour le calcul algébrique. D'un point de vue théorique d'abord puisqu'il a fallu attendre 1993 pour connaître le nombre maximal de solutions complexes du problème [82], [106] ou [96]. Ce nombre est devenu calculable en pratique par des méthodes exactes en 1994 pour des robots « génériques » (le système était résolu modulo un nombre premier) [48]. Le calcul du nombre de solutions réelles est resté longtemps un problème ouvert traité strictement numériquement. Dans **FR**-[113], **FR**-[112], le calcul du nombre de solutions réelles devient pour la première fois effectif de façon certifiée modulo plusieurs heures (voir jours) de calcul. Ce résultat avait toutefois permis d'exhiber un robot général admettant 24 solutions, ce qui représentait à l'époque le maximum connu. C'est un processus d'optimisation ad-hoc numérique (non certifié) [38] qui a permis d'exhiber un robot parallèle admettant 40 solutions a priori réelles. Dans **FR**-[49], une nouvelle modélisation du problème, couplée aux méthodes générales de résolution de systèmes zéro-dimensionnels présentées dans ce document, permettent de certifier ce résultat mais surtout de résoudre le problème géométrique direct de n'importe quel robot en des temps actuellement voisins de la seconde avec un résultat certifié.

- *Systèmes de dimension positive*. Deux applications ont servi d'objectifs : la synthèse de bancs de filtres non séparables en dimension 2 et le problème d'interpolation de Birkhoff. Dans les deux cas, les processus de résolution nécessitaient un algorithme permettant de décider si un système algébrique de dimension quelconque admettait ou non des solutions réelles. La première application, que l'on retrouve très partiellement dans **FR**-[113], est à l'origine de l'investissement sur la résolution des systèmes généraux (tester si une variété algébrique admet ou non des solutions réelles) mais la difficulté à été contournée dans **FR**-[45] grâce à une modélisation différente. La seconde application, que l'on trouve également dans **FR**-[113] partiellement résolue « à la main », à servi à valider l'approche « points critiques » (stratégies gigognes) comparativement aux techniques « récursives » (CAD et suites de Sturm), mais surtout d'opérer un virage par rapport aux algorithmes que l'on retrouve dans [13]. En effet, l'article **FR**-[119] illustre clairement que l'approche mathématique préconisée dans [13] consistant à se ramener systématiquement à un problème plus simple (déformations infinitésimales, sommes de carrés, etc.) montre rapidement ses limites en pratique alors que la stratégie proposée dans **FR**-[7] fournit une boite noire permettant d'augmenter l'état de l'art sur le problème.
- *Systèmes dépendant de paramètres*. L'application « fil rouge » pour ce type de problème à été la classification d'une famille de robots série à 3 degrés de liberté selon leur caractère cuspidal. Ce problème est résolu de manière ad-hoc dans **FR**-[33] pour le cas d'un robot dépendant de 3 paramètres de conception grâce à un changement de variable obtenu par tâtonnements successifs et une utilisation manuelle de la structure des équations. Ce résultat a été étendu ensuite au cas de robots dépendant de 4 paramètres dans [32], utilisant des résultats partiels de **FR**-[86]. Cette dernière référence propose une méthode générale et complète de résolution de systèmes dépendant de paramètres permettant maintenant de résoudre automatiquement ce type de problème.

La chronologie des résultats

Les résultats de **FR**-[113] étaient essentiellement dédiés aux systèmes de dimension 0, les travaux relatifs aux autres cas étant essentiellement du domaine de la prospective et reliés à la résolution ad-hoc de certaines applications « fil-rouge ». L'objectif de ce travail était d'une part d'étudier la viabilité de méthodes non récursives sur les variables et le bien fondé des bases de Gröbner comme pré-requis éventuel, d'autre part de fournir des boites noires solides pour valider l'approche « méthodes de points critiques » proposée dans [11], [12] et que l'on peut aujourd'hui trouver dans [13]. L'effort a porté essentiellement sur l'exploitation de la structure de l'anneau quotient pour compter et isoler les solutions d'un système ou simplement éliminer des variables.

On trouve dans **FR**-[113] une première version d'un algorithme clé pour ce type de systèmes : la Représentation Univariée Rationnelle (RUR): l'objet calculer permet de se ramener systématiquement à l'étude de polynômes en une variable. Les résultats de **FR**-[113] ont été consolidés et améliorés (par exemple, abandon des fonctions multi-symétriques) dans **FR**-[114] puis **FR**-[117].

L'application « modèle géométrique direct des robots parallèles » à mis en évidence le manque d'outils efficaces pour la résolution des polynômes en une variable. En effet, les outils courants du moment (essentiellement les suites de Sturm) ne permettaient plus de résoudre les problèmes en une variables relatifs à l'utilisation de RURs comme objet intermédiaires. Plusieurs travaux, en particulier ceux visant à améliorer les calculs de décomposition cylindrique, ont montré que les stratégies basées sur la règle des signes de Descartes pouvaient être une solution pour toute une gamme de problèmes. Cette substitution de « boite noire » pour le traitement des problèmes en une variable a permis de changer rapidement de classe d'exemples traitables et de nouveaux problèmes sont apparus, essentiellement liés au contrôle de l'occupation mémoire des algorithmes. L'algorithme RUR a été complètement modifié pour minimiser le nombre d'objets stockés **FR**-[114],**FR**-[117] et qu'une version « optimale en mémoire » ainsi qu'une variante utilisant intensivement de l'arithmétique flottante **FR**-[110] de l'algorithme connu sous le nom d'algorithme de Uspensky à été proposée dans **FR**-[121].

Pariant sur la mise au point d'un algorithme efficace pour la résolution des systèmes zéro-dimensionnels, les méthodes de points critiques ont été étudiées parallèlement (en particulier grâce au travail réalisé pendant la thèse de Mohab Safey El Din [123]). L'idée initiale a été de suivre la feuille de route proposée dans [11] (voir [13] pour une vue générale), les premières investigations ont donc eu pour objectif le calcul d'un point par composante connexe de variétés réelles définies par une unique équation avec pour objectif premier de minimiser les recours aux déformations infinitésimales alourdissant les calculs au point de les rendre infaisables. Dans cette famille de méthodes, l'idée principale est de trouver une fonction polynomiale atteignant ses extrema sur chaque composante connexe. Si le nombre de ces extrema est fini, le calcul d'un point par composante connexe se ramène alors à la résolution de systèmes zéro-dimensionnels. Selon la fonction polynomiale choisie, cette stratégie générale peut induire quelques contraintes (compacité, lissité, etc.) sur la variété étudiée. Pour les fonctions de type « projection », ces contraintes sont contournées en effectuant des déformations infinitésimales nécessitant a priori de travailler en pratique dans le corps des séries de Puiseux. Comme proposé dans **FR**-[118], il est cependant possible d'éviter l'utilisation d'arithmétiques aussi lourdes dans un certain nombre de cas, mais le passage au problème d'ordre supérieur (variétés définies par un nombre arbitraire d'équations) est resté impossible en pratique à cause par exemple de l'élévation artificielle des degrés des équations (par exemple remplacement d'un système par la somme des carrés de ses équations). Il est montré dans **FR**-[115] que l'application brutale d'algorithmes de bonne complexité théorique tels ceux de [13] se heurtera de toutes façons à des problèmes pratiques a priori inextricables. L'article **FR**-[7] représente un virage puisqu'il s'écarte de ce type de stratégie en évitant systématiquement les artifices mathématiques (déformations infinitésimales), avec pour conséquence d'abandonner le passage obligé par le cas des hypersurfaces pour se concentrer directement sur celui des systèmes quelconques; l'approche est validée en pratique sur l'application « problème d'interpolation de Brikhoff » dans **FR**-[119].

La délégation de ce sujet de recherche (début des années 2000), désormais avantageusement développé et étendu par Mohab Safey El Din (voir [125] ou [126]), a permis de se concentrer sur la résolution des systèmes dépendant de paramètres. L'application « fil rouge » associée (« robots cuspidaux ») prenait en défaut l'état de l'art des solutions praticables : le problème se ramène à l'étude d'un système en 7 variables (3 indéterminées et 4 paramètres) sur lequel aucune méthode connue ne pouvait s'appliquer. Une solution « ad-hoc » basée sur un changement de variables intuité expérimentalement et exploitant la structure très particulière d'une réécriture du système a permis de résoudre le problème lorsque l'un des paramètres est nul **FR**-[33]. Ce cas particulier, jugé d'importance par les spécialistes, à été à l'origine de quelques études spécialisées [103], [144]. L'analyse de quelques calculs aux caractéristiques similaires ([85],[74],[84] ou encore [83]) a montré que la résolution de tels systèmes induisait directement ou indirectement le calcul d'un certain nombre de sous-variétés de l'espace des paramètres du problème. Une variété optimale (variété discriminante) est définie dans **FR**-[86]. Son calcul permet d'éliminer en une seule passe toutes les indéterminées du problème, ramenant celui-ci à l'étude du complémentaire d'une sous-variété de l'espace des paramètres ou, plus précisément, de ses composantes connexes. Dans le cas réel, cette stratégie utilise donc les « méthodes de points critiques » évoquées ci-dessus pour obtenir des résultats qualitatifs et supplée avantageusement les méthodes basées sur la décomposition cylindrique algébrique. Une version restreinte de l'algorithme général est utilisée dans la thèse de Solen Corvez [32] pour résoudre le cas général de l'application « robots cuspidaux » et sa complexité est maintenant mieux connue [93] (premier résultat publié de la thèse de Guillaume Moroz).

Chapitre 2

Polynômes en une variable

2.1 Introduction

Cette étude sur les méthodes d'isolation de polynômes en une variable à coefficients rationnels a été motivée par les applications nécessitant la résolution de systèmes zéro-dimensionnels. Sur ce sujet, l'effort a été initialement porté sur l'élimination de variables et en particulier la Représentation Univariée Rationnelle (RUR) permettant de ramener l'étude de tout système zéro-dimensionnel à celle de polynômes en une variable. Dans le schéma global de résolution enchaînant une réécriture du système (par exemple un calcul de base de Gröbner) puis le calcul d'une RUR puis la résolution du problème en une variable (voir chapitre 3), la dernière phase, initialement résolue en utilisant des stratégies existantes (basées sur des calculs de suites de Sturm-Habicht), est rapidement devenue bloquante. Le tableau qui suit, détaillé et actualisé dans la section 2.8, donne le poids de la résolution du premier polynôme d'une RUR par rapport au temps de calcul de celle-ci, à partir du système initial, l'unité de temps étant le temps du calcul complet de la RUR (Base de Gröbner + RUR) : par exemple, pour l'exemple `katsura7`, l'isolation du premier polynôme de la RUR prend 66 fois plus de temps que le calcul de RUR, pour l'exemple `katsura8`, elle est l'étape bloquante du processus.

Exemple	Sturm-Habicht	Nb. Real	Degree
Fabrice24	3.9	24	40
camera1s	0.8	16	20
cpdm5	46.4	43	163
cyclic6	9.8	24	156
cyclic7	∞	56	924
d1	3.6	16	48
delta2	9.1	8	40
des18_3	4.3	6	46
des22_24	1.7	10	42
ducos7_3	5.7	168	168
hexa2	8.6	8	40
hexaglide	13.3	8	36
i1	∞	16	66
ilias_k_2	0.7	14	72
ilias_k_3	0.7	18	84
ingersoll	8.7	6	40
katsura6	12.7	32	64
katsura7	66.0	44	128
katsura8	∞	84	256
katsura9	∞	120	512

Il existe plusieurs méthodes exactes pour le comptage ou l'isolation des zéros réels d'un polynôme en une variable à coefficients rationnels. La plus simple est certainement la suivante : on borne la valeur absolue des racines par un rationnel $M(P)$ ainsi que la distance minimale entre 2 racines par un autre rationnel $\mathrm{Sep}(P)$ et on remplace, si nécessaire, le polynôme étudié par sa partie sans facteurs carrés. On calcule ensuite $l = \left\lceil \frac{M(P)}{S(P)} \right\rceil$ rationnels $a_1 < ... < a_l$ tels que $|a_{i+1} - a_i| < \mathrm{Sep}(P)$ et on évalue la partie sans facteurs carrés du polynôme étudié en les a_i : l'intervalle $[a_i, a_{i+1}]$ contient une (unique) racine de p si et seulement si $p(a_i)p(a_{i+1}) \leq 0$. Le comportement de ce type de méthode est essentiellement conditionné par la qualité des bornes $M(P)$ et $S(P)$ que l'on peut calculer en fonction des coefficients du polynôme étudié, les bornes les plus connues étant :

Proposition 2.1. *Soit $P = \sum_{i=0}^{d} a_i X^i \in \mathbb{Q}[X]$ tel que $a_n = 1$ et $a_0 \neq 0$, et α une racine de P.*

- *[23] $|\alpha| < 1 + m\,a\,x_{i=0}^{d}(|a_i|) = M(P)$;*
- *[89] Si α et β sont deux racines distinctes de P, alors $|\alpha - \beta| \geq \sqrt{\frac{3}{d^d + 2}} \cdot \frac{1}{||P||_2^{d-1}} = \mathrm{Sep}(P)$, avec $||P||_2 = \sqrt{\sum_{i=0}^{d} a_i^2}$;*

Même si des bornes plus fines existent, celles-ci ne diminuent pas la complexité théorique de l'algorithme qui est clairement exponentielle en le degré du polynôme étudié.

A quelques exceptions près, les méthodes exactes modernes réputées efficaces ne découpent pas aussi brutalement l'intervalle $]-M, M[$ mais réalisent des bissections successives. Au final elles calculent le début du développement binaire des racines recherchées. On peut toutefois pointer l'algorithme [3] calculant un développement en fractions continues. Il ne fait pas partie de l'étude résumée ici pour deux raisons : la première est que sa complexité est mal maîtrisée, même si quelques conjectures laisseraient penser que celle-ci est comparable à celles que nous obtenons, la deuxième est qu'expérimentalement il ne semble pas apporter de valeur ajoutée en termes de temps de calculs, sauf sur certains exemples très particuliers ou les racines ont un développement en fractions continues s'avérant beaucoup plus simple que leur développement binaire (par exemple les polynômes de Mignotte).

Les principes de base de ces familles d'algorithmes sont rappelés dans la section 2.2 ainsi que deux variantes historiques. Le premier algorithme s'appuie sur un calcul de suites de Sturm [134] et constitue une référence, d'une part parce que sa complexité est depuis relativement longtemps bien maîtrisée [35], d'autre part parce qu'il s'agissait de la technique la plus standard au départ du travail exposé dans ce chapitre. Le deuxième algorithme [27] utilise la règle des signes de Descartes [37] et à servi de base pour l'étude proposée dans ce qui suit : il est utilisé depuis longtemps dans des méthodes de Décomposition Cylindrique Algébrique [30]. Un historique intéressant sur l'histoire de cette méthode peut être trouvé dans [2], donnant en particulier les contributions effectives de [139], [138] ou encore [36].

La section 2.3 présente une homogénéisation de l'ensemble des méthodes de bissection basées sur la règle de Descartes comme par exemple celle de [77]. Cette formalisation permet en particulier d'isoler les fonctions sensibles conditionnant le comportement de ce type d'algorithmes (ordre dans lequel les calculs sont effectués) ainsi que ses défauts (occupation mémoire par exemple).

La section 2.4 présente une variante, toujours basée sur le principe de bissection et la règle de Descartes, optimisant l'occupation mémoire sans sacrifier la rapidité d'exécution. En effet, modulo quelques choix importants pour les opérations de base (essentiellement la substitution $X \longrightarrow X + 1$), ce type de méthode c'est immédiatement avéré d'une efficacité pratique remarquable au point de n'être bloqué, sur certaines classes d'exemples, que par une occupation mémoire excessive.

La section 2.5 s'attache au cas des polynômes exprimés dans une base de Bernstein au lieu de la base de monômes classique (cas courant dans certains domaines comme par exemple en géométrie algorithmique). Des instances de l'algorithme général décrit dans la section 2.3 sont proposés, reprenant essentiellement la méthode de [79], réactualisée dans [99].

Les complexités binaires des différents algorithmes présentés sont établies dans la section 2.6, et la section 2.7 montre que l'on peut utiliser une arithmétique en précision fixe pour accélérer les calculs (pas de croissance de coefficients) tout en certifiant le résultat final.

Quelques temps de calcul illustrant les principales propriétés des divers algorithmes sont proposés dans la section 2.8 et la section 2.9 énumère quelques applications relativement directes de ces méthodes comme le calcul de signes ou l'évaluation de polynômes en des nombres algébriques réels.

2.2 Généralités sur les méthodes à base de bissection

Les méthodes modernes d'isolation exactes sont majoritairement basées sur une bissection de $[-M; M]$: si on se munit d'une fonction $V(]\alpha, \beta])$ capable de compter le nombre de zéros de P dans un intervalle $]\alpha, \beta]$, le corps de la routine d'isolation est la fonction suivante :

Algorithme `BissectionSimple(`$]\alpha, \beta], V$`)`
`Si` $V(]\alpha, \beta]) > 0$ `alors`
`Si` $V(]\alpha, \beta]) = 1$, $\mathrm{Res} := \{]\alpha, \beta]\}$;
`Sinon Res:=BissectionSimple(`$]\alpha, \frac{\alpha+\beta}{2}]$`) ∪ Bissection(`$]\frac{\alpha+\beta}{2}, \beta]$`)`
`Retourner(`Res`)`

La fonction qui suit décrit alors un algorithme d'isolation complet, modulo la donnée d'une fonction V permettant de compter exactement le nombre de solutions réelles d'un polynôme de $\mathbb{Q}[X]$ dans un intervalle $]\alpha, \beta]$. Ici, on se ramène systématiquement au cas d'un polynôme ayant toutes ses racines dans $[0, 1]$ de sorte à ne plus s'encombrer la borne $M(P)$ dans les études qui suivront :

Algorithme `Isolate(`P`,`V`)`
Tmp:={}
$M := M(P)$
$Q(X) := (2M)^d P(2MX - M)$
`Si` $Q(0) = 0$ Tmp:=`{[0,0]}`; $Q(X) := \frac{Q(X)}{X}$;
Tmp:=Tmp ∪ `BissectionSimple(`$]0, 1], V$`);`
`Retourner(`$\{(2M\alpha - M, 2M\beta - M), (\alpha, \beta) \in \mathrm{Tmp}\}$`)`

Mis à part les changements de variables opérés dans la fonction Isolate, on peut résumer le comportement de ce type de méthode par le parcours d'un arbre binaire dont les noeuds sont des intervalles de la forme $I_{k,c} =]\frac{c}{2^k}, \frac{c+1}{2^k}]$, visiter un noeud consistant à calculer le nombre de zéros réels du polynôme dans l'intervalle correspondant. Pour k fixé, le nombre d'intervalles $I_{c,k}$ visités ne peut dépasser le nombre de zéros du polynôme. De plus, la profondeur maximum de l'arbre (valeur maximale de k lors des appels à la fonction Bissection) est majorée par $\frac{1}{2^{k_{\max}}}$, $k_{\max}$ étant le plus petit entier tel que $\frac{1}{2^{k_{\max}}} < \mathrm{Sep}(P)$, puisque pour tout entier $k \geq k_{\max}$, le nombre de zéros de P dans $I_{k,c}$ est nécessairement 0 ou 1. Ceci donne un nombre total de noeuds visités qui est polynomial (au plus quadratique) en le degré de P et en la taille de ses coefficients (aux termes en log près) et induit un algorithme de complexité polynomiale en ces mêmes quantités dès lors que l'algorithme implantant la fonction V l'est.

2.2.1 Utilisation de suites de Sturm

Une instance facile à mettre en oeuvre utilise les suites de Sturm (ou Sturm-Habicht) pour implanter la fonction V. Pour rappel, une suite de Sturm associée à un polynôme P à coefficients dans un anneau est une séquence d'au plus d polynômes $\mathrm{Stu}_0(P), ..., \mathrm{Stu}_l(P)$ à coefficients dans ce même anneau. Sans rentrer dans les détails de leur définition formelle, la principale propriété de ces suites est donnée par le théorème suivant :

Théorème 2.2. *[134] Soit* $P \in \mathbb{Q}[X]$ *et* $\mathrm{Stu}_0(P), ..., \mathrm{Stu}_l(P)$ *une suite de Sturm associée à P. Pour toute suite de scalaires non nuls* $[a_0, ..., a_d]$*, on définit son nombre de changements de signes* $\mathrm{Var}(a_0, ..., a_d)$ *par récurrence :* $\mathrm{Var}(a_0) = \mathrm{signe}(a_0)$ *et* $\forall j > 0$, $\mathrm{Var}(a_0, ..., a_j) = \mathrm{Var}(a_0, ..., a_{j-1}) + 1$ *si* $\mathrm{signe}(a_j a_{j-1}) = -1$, $\mathrm{Var}(a_0, ..., a_j) = \mathrm{Var}(a_0, ..., a_{j-1})$ *sinon . Alors,* $\forall a < b$ *le nombre de zéros de P dans* $[a, b]$ *est égal à* $\mathrm{Var}(\mathrm{Stu}_0(P)(b), ..., \mathrm{Stu}_l(P)(b)) - \mathrm{Var}(\mathrm{Stu}_0(P)(a), ..., \mathrm{Stu}_l(P)(a))$.

Le moyen le plus simple de calculer une suite de Sturm est de construire la suite des restes signés associée à P et à sa dérivée, en posant $\mathrm{Stu}_0(P) = P$, $\mathrm{Stu}_1(P) = P'$, $\mathrm{Stu}_2(P) = -\mathrm{reste}(P, P')$, ..., $\mathrm{Stu}_i(P) = -\mathrm{reste}(\mathrm{Stu}_{i-1}(P), \mathrm{Stu}_{i-2}(P))$, ... jusqu'à obtenir un polynôme de degré nul, $\mathrm{reste}(P, Q)$ désignant le reste de la division Euclidienne de P par Q.

Modulo le calcul d'une suite de Sturm associée à P, la fonction V nécessaire à l'application des algorithmes décrits plus haut se résume alors en l'évaluation de cette suite en 2 points. L'algorithme global d'isolation est alors clairement polynomial tant en complexité arithmétique qu'en complexité binaire.

2.2.2 Utilisation de la règle des signes de Descartes

Au lieu d'utiliser un calcul exact pour V, l'idée est d'utiliser un calcul plus rapide mais ne donnant qu'une borne du nombre de zéros de P dans un intervalle $]\alpha, \beta]$. Bien que ce type de stratégie apparaisse depuis longtemps dans la littérature [139], [138], [36], ce qui suit peut être attribué à Collins et Akritas [27](couplage avec la stratégie de Bissection, preuve de terminaison de l'algorithme). Le principe de la méthode s'appuie sur une règle très simple :

Lemme 2.3. *(Règle des signes de Descartes [37]). Soit* $P = \sum_{i=0}^{d} a_i X^i$ *un polynôme de* $\mathbb{R}[X]$ *sans facteurs carrés avec* $a_0 \neq 0$*. On définit* $[b_0, ..., b_l]$ *la suite des coefficients non nuls de P (avec pour convention de numérotation : si* $i < j$ *alors* $b_i = a_{i'}$ *et* $b_j = a_{j'}$ *avec* $i' < j'$*) et* $\mathrm{Var}(P) = \mathrm{Var}([b_0, ..., b_l])$ *en reprenant la définition du théorème 2.2. Alors* $\boldsymbol{V}(P) \cap \mathbb{R}^{+\star} = \{x \in \mathbb{R}^{+\star}; P(x) = 0\} \leq \mathrm{Var}(P)$ *et, de plus,* $\boldsymbol{V}(P) \cap \mathbb{R}^{+\star} = \mathrm{Var}(P) \bmod 2$.

Si l'on déroule le premier appel à la fonction BISSECTIONSIMPLE introduite en début de cette section, on commence par calculer le nombre de racines de P dans $]0, 1]$ (puis, le cas échéant, on regarde les intervalles $]0, 1/2]$ et $]1/2, 1]$). L'opération de base introduite par Collins et Akritas consiste à borner ce nombre en composant la règle de Descartes, en utilisant des transformations simples :

Lemme 2.4. *[27] Soit* $P = \sum_{i=0}^{d} a_i X^i$ *un polynôme de* $\mathbb{R}[X]$ *sans facteurs carrés tel que* $P(0) \neq 0$ *et* $P(1) \neq 0$*. Alors* $\boldsymbol{V}(P) \cap]0, 1[\leq \mathrm{Var}((X+1)^d P(\frac{1}{1+X}))$ *et* $\boldsymbol{V}(P) \cap]0, 1[= \mathrm{Var}((X+1)^d P(\frac{1}{1+X})) \bmod 2$.

D'après le lemme 2.4, si la borne calculée est 0 (resp. 1), P n'a pas de racines (resp. a exactement une racine) dans $]0, 1[$ et la bissection s'arrête (on traite séparément le cas ou $P(1) = 0$). Dans le cas contraire, on explore les intervalles $]0, 1/2]$ et $]1/2, 1]$ l'un après l'autre en remarquant que les racines de P dans $]0, 1/2]$ sont en bijection avec celles de $Q_1 = P(X/2)$ dans $]0, 1]$ et que les racines de P dans $]1/2, 1]$ sont en bijection avec celles de $Q_2 = Q_1(X+1)$ dans $]0, 1]$. Pour résumer, l'opération de bissection proposée par Collins et Akritas est la suivante :

Algorithme `BissectionColAkr`$(]\alpha, \beta], Q)$
Res $= \{\}$
`Si` $\mathrm{Var}((X+1)^d Q(\frac{1}{1+X})) > 0$ `alors`
  `Si` $\mathrm{Var}((X+1)^d Q(\frac{1}{1+X})) = 1$, Res $:=$ Res $\cup \{]\alpha, \beta]\}$;
`Sinon`
  Res $:=$ Res $\cup$ `BissectionColAkr`$(]\alpha, \frac{\alpha+\beta}{2}], 2^d Q(X/2))$
    $\cup$ `BissectionColAkr`$(]\frac{\alpha+\beta}{2}, \beta], 2^d Q(\frac{X+1}{2}))$
  `Si` $Q(1/2) = 0$ `alors` Res $:=$ Res $\cup \{\frac{\alpha+\beta}{2}\}$
`Retourner`(Res)

Sans résultat supplémentaire, on ne peut garantir la terminaison de l'algorithme, le calcul du nombre de zéros dans un intervalle ayant été remplacé par une borne. Les deux théorèmes suivants garantissent que si $|\alpha - \beta|$ est suffisamment petit, alors $\mathrm{Var}(Q(\frac{1}{1+X})) = 0$ ou 1, prouvant que l'algorithme global terminera toujours :

Théorème 2.5. *[139] Si $P \in \mathbb{R}[X]$ est un polynôme sans facteurs carrés admettant exactement une racine réelle dans]0,1[et tel que toutes ses racines non réelles soient à l'extérieur de 2 disques unitaires de centres $(0,0)$ et $(1,0)$ dans le plan complexe, alors $\mathrm{Var}(P(\frac{1}{1+X})) = 1$.*

Théorème 2.6. *[31] Si $P \in \mathbb{R}[X]$ est un polynôme sans facteurs carrés n'ayant aucune racine réelle dans le disque de rayon $1/2$ et de centre $(1/2, 0)$, alors $\mathrm{Var}(P(\frac{1}{1+X})) = 0$.*

2.3 Une description unifiée des algorithmes de bissection

Les algorithmes présentés dans la section précédente visitent les intervalles $I_{k,c} =]\frac{c}{2^k}, \frac{c+1}{2^k}]$ du processus de Bissection, que l'on peut voir comme les noeuds d'un arbre binaire. L'ordre dans lequel ces intervalles sont visités dans le cas de l'algorithme de Collins/Akritas a une importance capitale puisqu'il permet d'effectuer des opérations simples (et rapides) : l'essentiel des calculs étant constitué de substitutions du type $X \longrightarrow X/2$ et $X \longrightarrow X + 1$. Mais la structure récursive de l'algorithme engendre une occupation en mémoire qui peut s'avérer bloquante. En effet, l'exploration de l'intervalle $I_{k,c} =]\frac{c}{2^k}, \frac{c+1}{2^k}[$ nécessite de transformer un polynôme calculé lors de l'exploration d'un intervalle de la forme $I_{k-1,c'}$. Aucune ne ces transformations ne pouvant être destructive, l'exploration de l'intervalle $I_{k,c}$ nécessite le stockage de tous les polynômes calculés aux étapes précédentes, c'est à dire jusqu'à $k_{\max}$ polynômes ou $k_{\max}$ est majoré par $O(\log_2(\frac{1}{\mathrm{Sep}(P)}))$. Pour un polynôme de degré d avec des coefficients de taille t, le nombre de polynômes est en $\tilde{O}(d\,t)$, borne dont on s'approche en pratique dans le cas de polynômes ayant des racines très proches (c'est par exemple le cas de polynômes de Mignotte). Pour certaines applications, il faut avoir à l'esprit que les variantes modernes permettent de traiter des exemples ou d peut atteindre des valeurs de l'ordre de 10 000 pour t de l'ordre de 1000 : la méthode ne se trouvera bloquée que par une occupation mémoire prohibitive.

Ce problème est connu de longue date et a engendré la mise au point de plusieurs variantes dont la plus aboutie est certainement celle présentée dans [77]. Le principe de ces évolutions de l'algorithme est de visiter les intervalles $I_{k,c}$ dans un autre ordre. Par exemple dans [77], c'est un parcours « en largeur » qui est effectué sur l'arbre binaire représentant l'ensemble de ses intervalles (au lieu d'un backtracking pour l'algorithme de Collins/Akritas), majorant alors le nombre de polynôme stockés par le nombre d'intervalles $I_{k,c}$ pour k fixé, par le degré du polynôme étudié.

Pour étudier et optimiser ce type de méthodes, l'idée de **FR**-[121] est de dissocier l'arbre de calcul (intervalles $I_{k,c}$), l'ordre dans lequel sont parcourus ses noeuds (stratégie) et les opérations de base (transformations sur les polynômes et calculs de bornes).

Le formalisme de **FR**-[121] est légèrement généralisé dans ce qui suit pour englober l'utilisation de suites de Sturm ou de tout autre procédé permettant de borner ou compter les racines d'un polynôme donné dans un intervalle.

Définition 2.7. *Soit P un polynôme de $\mathbb{R}[X]$ de degré d, k et c deux entiers positifs ou nuls tels que $c < 2^k$. On définit :*

- $I_{k,c} =]\frac{c}{2^k}, \frac{c+1}{2^k}]$
- $\mathrm{RootBound}(\mathrm{Node}_{k,c}(P))$ *: une fonction calculant une borne sur le nombre de racines de P dans $I_{k,c}$, $\mathrm{Node}_{k,c}(P)$ étant un raccourci désignant l'ensemble de ses paramètres associés à un noeud.*

- $\text{Internal}(P) = \{(k,c)\colon P(\frac{c}{2^k}) \neq 0, \text{RootBound}(P, I_{k,c}) > 1\}$,
- $\text{Tree}(P) = \{(0,0)\} \cup \{(k,c)\colon (k-1, \lfloor c/2 \rfloor) \in \text{Internal}(P)\}$,
- $\text{Exact}(P) = \{(k,c) \in \text{Exact}(P)\colon P_{k,c}(\frac{c}{2^k}) = 0\}$,
- $\text{Isol}(P) = \{(k,c) \in \text{Tree}(P)\colon P(\frac{c}{2^k}) \neq 0, \text{RootBound}(P, I_{k,c}) = 1\}$,
- $\text{Lost}(P) = \{(k,c) \in \text{Tree}(P)\colon P(\frac{c}{2^k}) \neq 0, \text{RootBound}(P, I_{k,c}) = 0\}$.

Les variantes de l'algorithme de bissection diffèrent par

- la façon dont sont représentés les noeuds de $\text{Tree}(P)$. Par exemple dans le cas de l'utilisation de suites de Sturm, supposant une suite de Sturm de P pré-calculée, un noeud sera simplement représenté par le couple (k,c). Dans le cas de l'algorithme de Collins/Akritas, il faudra lui adjoindre le polynôme Q sur lequel on calcule $\text{Var}((X+1)^d Q(\frac{1}{1+X}))$;
- l'ordre dans lequel sont visités les noeuds : dans les deux exemples de la section précédente, il s'agit d'un backtracking, mais dans l'algorithme [77], il s'agit d'un parcours « en largeur »;
- la façon dont est calculée la borne sur le nombre de racines dans $I_{k,c}$: évaluation de la suite de Sturm ou utilisation de la règle de Descartes;
- la façon dont sont calculés les successeurs d'un noeud. Dans le cas d'utilisation d'une suite de Sturm, il s'agit simplement de calculer $I_{k+1,2c}$ et $I_{k+1,2c+1}$, mais dans le cas de l'algorithme de Collins Akritas, il faut calculer en plus les 2 polynômes sur lesquels il faudra appliquer la règle de Descartes.

Les deux premiers points conditionnent l'occupation mémoire de l'algorithme, les 3 derniers son efficacité (le deuxième point est en particulier critique pour ces 2 critères). Ces éléments sont donnés en paramètres de l'algorithme général présenté ci-dessous sous la forme suivante :

- $<$ désigne un ordre sur $\mathbb{N}^2$: il est donné en paramètre de la fonction $\text{getNode}(T, <)$ qui permet de choisir le plus petit élément d'in sous-ensemble T de $\text{Tree}(P)$;
- $\text{Node}_{k,c}(P)$ est simplement la représentation d'un noeud de $\text{Tree}(P)$ modélisant l'intervalle $I_{k,c}$, $\text{initTree}(P)$ ne sert qu'à des fins cosmétiques pour « initialiser » $\text{Tree}(P)$ en calculant $\text{Node}_{0,0}(P)$;
- $\text{RootBound}(\text{Node}_{k,c}(P))$ est n'importe quelle fonction permettant de borner le nombre de zéros de P dans l'intervalle $I_{k,c}$;
- $\text{addSuccessor}(\text{Node}_{k,c}(P))$ calcule simplement $\text{Node}_{k+1,2c}(P)$ et $\text{Node}_{k+1,2c+1}(P)$, c'est à dire les 2 noeuds fils de $\text{Node}_{k,c}(P)$ en testant au passage le cas particulier ou $P(\frac{c}{2^k}) = 0$.

Algorithme `GenericBissection`$(P, \text{getNode}, <, \text{addSuccessor}, \text{RootBound})$
$\text{Exact}(P) \leftarrow \emptyset$, $\text{Isol}(P) \leftarrow \emptyset$, $T \leftarrow \text{Node}_{0,0}(P)$
`While` $T \neq \emptyset$ `do`
$\text{Node}_{k,c}(P) \leftarrow \text{getNode}(T, <)$
$s \leftarrow \text{RootBound}(\text{Node}_{k,c}(P))$
`If` $s = 1$ `then` $\text{Isol}(P) \leftarrow \text{Isol}(P) \cup \{(k,c)\}$
`If` $s > 1$ `then` $(T, \text{Exact}(P)) \leftarrow \text{addSuccessors}(\text{Node}_{k,c}(P))$

Pour spécialiser l'algorithme GENERICBISSECTION de sorte à ce qu'il implante la stratégie à base de suites de Sturm décrite dans la section précédente, il suffit de considérer les paramètres suivants :

SturmBissection

- $\text{Node}_{k,c}(P)$ est représenté par le couple (k, c) et une suite de Sturm de P. Remarque : tous les noeuds peuvent (et doivent en pratique) partager la même suite de Sturm.
- RootBound($\text{Node}_{k,c}(P)$) consiste simplement à évaluer la suite de Sturm commune aux noeuds en les points $\frac{c}{2^k}$ et $\frac{c+1}{2^k}$
- addSuccessor($\text{Node}_{k,c}(P)$) consiste simplement à évaluer P en $\frac{c}{2^k}$ pour tester si il s'annule en ce point (auquel cas on a trouvé une racine exacte) et à construire, les noeuds fils de $\text{Node}_{k,c}$(P), c'est à dire essentiellement calculer $k+1, 2\,c$ et $2\,c+1$.
- La fonction $<$ peut être quelconque, mais si l'on veut rester fidèle à l'algorithme décrit dans la section précédente, on prend l'ordre $(k, c) <_{\text{back}} (k', c')$ si et seulement si $\frac{c}{2^k} < \frac{c'}{2^{k'}}$ ou ($\frac{c}{2^k} = \frac{c'}{2^{k'}}$ et $k < k'$).

De la même manière, si on veut spécialiser cet algorithme général pour décrire l'algorithme de Collins/Akritas, il suffit de considérer les paramètres suivants (**FR**-[121]) :

CollinsAkritasBissection

- $\text{Node}_{k,c}(P)$ est représenté par le couple (k, c) et un polynôme $P_{k,c}$, en prenant $P_{0,0} = P$.
- RootBound($\text{Node}_{k,c}(P)$) consiste à calculer $\text{Var}((X+1)^d P_{k,c}(\frac{1}{1+X}))$
- addSuccessor($\text{Node}_{k,c}(P)$) consiste à évaluer $P_{k,c}$ en $\frac{1}{2}$ (ou de manière équivalente P en $\frac{c}{2^k}$) (pour voir si $\frac{c}{2^k}$ est une racine exacte) et à construire les noeuds fils de $\text{Node}_{k,c}$(P), c'est à dire essentiellement calculer $k+1$, $2\,c$ et $2\,c+1$ et les polynômes $P_{k+1,2c} = 2^d P_{k,c}(\frac{X}{2})$ et $P_{k+1,2c+1} = P_{k+1,2c}(X+1)$
- La fonction $<$ est l'ordre $(k, c) <_{\text{back}} (k', c')$ si et seulement si $\frac{c}{2^k} < \frac{c'}{2^{k'}}$ ou ($\frac{c}{2^k} = \frac{c'}{2^{k'}}$ et $k < k'$).

Enfin, on peut remarquer (**FR**-[121]) que l'amélioration apportée dans [77] sur l'occupation mémoire se résume à remplacer l'ordre $<_{\text{back}}$ par l'ordre $(k, c) <_{\text{lex}} (k', c') \iff k < k'$ ou $(k = k' \text{ et } c < c')$ dans la modélisation de l'algorithme de Collins/Akritas :

KrandickBissection

- $\text{Node}_{k,c}(P)$ est représenté par le couple (k, c) et un polynôme $P_{k,c}$, en prenant $P_{0,0} = P$.
- RootBound($\text{Node}_{k,c}(P)$) consiste à calculer $\text{Var}((X+1)^d P_{k,c}(\frac{1}{1+X}))$
- addSuccessor($\text{Node}_{k,c}(P)$) consiste à évaluer $P_{k,c}$ en $\frac{1}{2}$ (ou de manière équivalente P en $\frac{c}{2^k}$) (pour voir si $\frac{c}{2^k}$ est une racine exacte) et à construire les noeuds fils de $\text{Node}_{k,c}$(P), c'est à dire essentiellement calculer $k+1$, $2\,c$ et $2\,c+1$ et les polynômes $P_{k+1,2c} = 2^d P_{k,c}(\frac{X}{2})$ et $P_{k+1,2c+1} = P_{k+1,2c}(X+1)$
- La fonction $<$ est l'ordre lexicographique $(k, c) <_{\text{lex}} (k', c') \iff k < k'$ ou $(k = k' \text{ et } c < c')$.

2.4 Un algorithme de Descartes efficace et optimal en mémoire

Comme évoqué en introduction de cette section, l'occupation mémoire induite par les algorithmes basés sur la règle de Descartes est un élément essentiel d'efficacité. Un moyen simple de corriger cet avatar est simplement de remarquer que le polynôme $P_{k,c}$ utilisé pour calculer une borne sur le nombre de racines de P dans l'intervalle $I_{k,c}$ s'exprime simplement en fonction de P dans le cas de l'algorithme de Collins/Akritas ou de Krandick : $P_{k,c} = 2^{kd} P(\frac{X+c}{2^k})$.

Cette remarque permet d'optimiser instantanément, pour ce qui est de l'occupation mémoire, ces deux stratégies en considérant les paramètres suivants pour l'algorithme générique :

NaiveBissection

- $\text{Node}_{k,c}(P)$ est représenté par le couple (k, c), et par le polynôme P étudié (commun à tous les noeuds).
- RootBound($\text{Node}_{k,c}(P)$) calcule $P_{k,c} = 2^{kd} P(\frac{X+c}{2^k})$ puis $\text{Var}((X+1)^d P_{k,c}(\frac{1}{1+X}))$
- addSuccessor($\text{Node}_{k,c}(P)$) consiste simplement à évaluer P en $\frac{c}{2^k}$ pour tester si il s'annule en ce point (auquel cas on a trouvé une racine exacte) et à construire, les noeuds fils de $\text{Node}_{k,c}$(P), c'est à dire essentiellement calculer $k+1, 2\,c$ et $2\,c+1$.
- La fonction $<$ est n'importe quel ordre su $\mathbb{N}^2$.

Cette variante n'impose que le stockage d'un seul polynôme, elle est donc optimale (ou presque puisqu'un deuxième polynôme est temporairement stocké) pour ce qui est de l'occupation mémoire. Par contre, elle s'avère plus coûteuse en opérations arithmétiques que les versions de Collins/Akritas et Krandick. Pour observer ce phénomène, il faut comparer le calcul des polynômes $P_{k,c}$ pour k et c fixés, les autres opérations étant en tout point semblables. Dans le cas des méthodes déduisant $P_{k+1,2c}$ et $P_{k+1,2c+1}$ de $P_{k,c}$. Le calcul de $P_{k+1,2c}$ est négligeable puisqu'il s'agit de simples décalages sur l'encodage binaire des coefficients ($P_{k+1,2c} = 2^d P_{k,c}(\frac{X}{2})$) soit un coût en $O(d)$ opérations arithmétiques. Le calcul de $P_{k+1,2c+1}$ est une translation simple (Taylor Shift) et l'on peut alors appliquer l'un des algorithmes proposés dans [72]. Le plus rudimentaire, qui ne fait appel qu'à des additions de scalaires est celui qui sera utilisé dans la suite : en pratique c'est le plus efficace. Son principe est simple : au lieu de calculer directement $Q(X+1)$ en développant le polynôme plus ou moins naïvement, on calcule ses polynômes de Horner $H_0, ..., H_d$. Pour rappel, le j-ième polynôme de Horner associe à $Q(X) = \sum_{i=0}^{d} a_i X^{d-i}$ est défini par $H_j(X) = \sum_{i=0}^{j} a_i X^{j-i}$, H_j et H_{j+1} étant liés par la relation $H_{j+1}(X) = X\,H_j(X) + a_j$. Ceci nous donne en particulier $\forall j > 0$, $H_j(X+c) = X\,H_{j-1}(X+c) + c\,H_{j-1}(X+c) + a_j$ et $H_d(X+c) = Q(X+c)$ d'où on déduit l'algorithme suivant ($Q[k]$ désigne le coefficient de X^k):

Algorithme `Translate(`P, c`)`

```
Q := P
For i=1 to d
   For k=d-i to d-1
       Q[k] := Q[k] + c Q[k+1]
Retourner(Q))
```

On peut observer que, dès lors que $c = 1$, cet algorithme n'utilise que des additions. Même en optimisant un peu plus le changement $X \longrightarrow X + c$, c'est à dire en calculant $Q(X + c) = Q(c\,(\frac{X}{c} + 1))$ de sorte à minimiser le nombre de multiplications, la méthode naïve proposée ci-dessus reste moins efficace en termes de temps de calcul pour 2 raisons :

- les calculs de $P_{k+1,2c}$ et $P_{k+1,2c+1}$ sont équi-coûteux;
- le coût des multiplications n'est pas négligeable;

Au lieu de calculer systématiquement $P_{k,c}$ à partir de P, l'idée proposée dans **FR**-[121] est de calculer $P_{k,c}$ à partir d'un autre polynôme de la forme $P_{k',c'}$, et de trouver un ordre dans lequel faire les calculs, minimisant le coût de ces transformations.

Proposition 2.8. *Étant donnés $P \in \mathbb{R}[X]$, $k, k', c, c' \in \mathbb{N}$ tels que $c < 2^k$ et $c' < 2^{k'}$, on a :*

$P_{k',c'} = 2^{d(k'-k)} H_{2^{k-k'}} \circ T_{2^{k-k'}c'-c}(P_{k,c})$ *et, inversement,* $P_{k,c} = 2^{d(k-k')}\, T_{c-2^{k-k'}c'} \circ H_{2^{k'-k}}(P_{k',c'})$, *avec comme convention :* $\forall \lambda \in \mathbb{Z}, T_\lambda(Q)(X) = P(X+\lambda), H_\lambda(Q)(X) = Q(\lambda X)$.

Le point clé pour obtenir un algorithme efficace est alors le résultat suivant :

Proposition 2.9. *En supposant que les noeud de l'arbre de calcul modélisant la bissection soient parcourus par ordre croissant pour l'ordre* $<_{\text{back}}$ *, c'est à dire* $(k,\, c) <_{\text{back}} (k',\, c')$ *si et seulement si* $\frac{c}{2^k} < \frac{c'}{2^{k'}}$ *ou (* $\frac{c}{2^k} = \frac{c'}{2^{k'}}$ *et* $k < k'$*), alors si* (k, c) *et* (k', c') *sont deux noeuds consécutifs,* $2^{k-k'}\, c' - c$ *vaut toujours* 0 *ou* 1*. En d'autres termes, si* (k', c') *et* (k, c) *sont deux noeuds consécutifs pour l'ordre* $<_{\text{back}}$ *avec* $(k,\, c) <_{\text{back}} (k',\, c')$*, alors* $P_{k',c'}$ *se déduit de* $P_{k,c}$ *par une homothétie de la forme* $X \longrightarrow 2^{\lambda} X$ *et éventuellement un Taylor shift (*$X \longrightarrow X + 1$*).*

On considère alors l'algorithme dont les spécifications sont les suivantes

RelativeBissection

- $\text{Node}_{k,c}(P)$ est représenté par le couple $(k,\, c)$ et par un polynôme de la forme $P_{k',c'}$ commun à tous les noeuds.
- RootBound($\text{Node}_{k,c}(P)$) calcule $P_{k,c} = 2^{d(k-k')} H_{2^{k'-k}} \circ T_{2^{k'-k}c-c'}(P_{k',c'})$, remplace le polynôme commun à tous les noeuds par $P_{k,c}$ puis calcule $\text{Var}((X+1)^d P_{k,c}(\frac{1}{1+X}))$
- addSuccessor($\text{Node}_{k,c}(P)$) consiste simplement à évaluer $P_{k,c}$ en $\frac{1}{2}$ pour tester si il s'annule en ce point (auquel cas on a trouvé une racine exacte : $\frac{c}{2^k}$) et à construire, les noeuds fils de $\text{Node}_{k,c}$(P), c'est à dire essentiellement calculer $k+1, 2\,c$ et $2\,c+1$.
- La fonction $<$ est l'ordre $<_{\text{back}}$.

On peut remarquer que l'algorithme ainsi décrit réalise exactement les mêmes opérations que l'algorithme de Collins/Akritas lors du passage d'un noeud (k, c) à un noeud (k', c') si $k' \geq k$. La différence principale est le passage d'un noeud $(k,\, c)$ à un noeud $(k',\, c')$ avec $k' < k$. Dans ce type de situation, on a forcément c modulo $2 = c'$ modulo $2 = 1$ si bien que le calcul effectué par l'algorithme de Collins/Akritas est $P_{k',c'} = T_1(P_{k',c'-1})$ alors que celui effectué par le nouvel algorithme est de la forme $P_{k',c'} = \frac{1}{2^{nh}} H_{2^h} T_1(P_{k,c})$, avec $h = k - k'$: on travaille donc alors avec des entiers plus gros. Même si le surcoût occasionné n'a pas d'influence significative sur le comportement global de l'algorithme (dans [78], il est montré que ce type de saut suffisamment faible en moyenne pour que ce calcul ne pèse pas sur la complexité globale de l'algorithme), on peut toutefois optimiser ce type de calcul en remarquant que, comme $P_{k',c'}$ est à coefficients entiers, alors les coefficients de $H_{2^h} T_1(P_{k,c})$ sont multiples de 2^{nh}. L'algorithme suivant (**FR**-[121]) est une variante de « Taylor shift » tenant compte du contenu, connu à l'avance, de $H_{2^h} T_1(P_{k,c})$:

Algorithme `HTranslate(`P`)`

$Q := P$
$\mu = \left\lceil \frac{d}{2^h} \right\rceil + h + 2$
`For i=0 to d` $Q[i] = \left\lfloor Q[i]\, 2^{\mu - h(d-i)} \right\rfloor$
`For i=1 to d`
  `For j=d-i to d-1`
    $Q[j] := Q[j] + \left\lfloor 2^{-h} Q[j+1] \right\rfloor$
`For i=0 to d` $Q[i] = \left\lfloor Q[i]\, 2^{-\mu} \right\rfloor$
`Retourner(`Q`)`

2.5 Le cas de polynômes exprimés dans la base de Bernstein

Dans certains types d'applications, essentiellement en géométrie algorithmique, les polynômes en une variable apparaissent exprimés dans une base de Bernstein et non dans la base usuelle des monômes. La forme la plus générale de base de Bernstein est donnée par :

Définition 2.10. *Soient α et β deux nombres réels, la base de Bernstein en degré d pour α et β est donnée par*

$$B_{d,i}[\alpha,\beta]=\binom{d}{i}\frac{(X-\alpha)^i(\beta-X)^{d-i}}{(\beta-\alpha)^d},\ i=0...d$$

Le résultat suivant est une adaptation des résultats de base utilisés par l'algorithme [79] (repris dans [99]) et une conséquence directe de la proposition 2.3 de **FR**-[98] : il établit le lien entre le test de variations de signes utilisé dans la méthode générale de bissection, c'est à dire $\mathrm{Var}((1+X)^dP_{k,c}(\frac{1}{1+X}))$ et le comptage du nombre de changements de signes dans la suite les coefficients de $P_{k,c}$ (resp. P) exprimé dans la base de Bernstein pour $\alpha=0$ et $\beta=1$ (resp. $\alpha=\frac{c}{2^k}$ et $\beta=\frac{c+1}{2^k}$), c'est à dire $\mathrm{Var}([c_0,...,c_d])$ si $P_{k,c}=\sum_{i=0}^d c_iB_{d,i}[0,1]$, (resp. $\mathrm{Var}([c_0,...,c_d])$ si $P=\sum_{i=0}^d c_iB_{d,i}[\frac{c}{2^k},\frac{c}{2^k}]$).

Proposition 2.11. $T_1(R_d(B_{d,i}[0,1]))=\binom{d}{i}X^{d-i}$, *ou* $R_d(P)=X^dP(1/X)$ *pour tout polynôme P de degré d. Ainsi, si $c_0,...,c_d$ désignent les coefficients de $P_{k,c}$ exprimés dans la base $B_{d,i}[0,1]$,* $\mathrm{Var}((1+X)^dP_{k,c}(\frac{1}{1+X}))=\mathrm{Var}([c_0,...,c_d])$. *De plus, à multiplication par 2^{dk} près (multiplicateur nécessaire pour obtenir des polynômes à coefficients entiers), les coordonnées de $P_{k,c}$ dans la base $B_{d,i}[0,1]$ sont égales aux coordonnées de P dans la base $B_{d,i}[\frac{c}{2^k},\frac{c+1}{2^k}]$.*

On peut, au moins dans le principe, déduire immédiatement une nouvelle instance de l'algorithme général de bissection :

BernsteinBissection

- $\mathrm{Node}_{k,c}(P)$ est représenté par le couple $(k,\ c)$, et par le polynôme P exprimé dans $B_{d,i}[\frac{c}{2^k},\frac{c+1}{2^k}]$.
- RootBound($\mathrm{Node}_{k,c}(P)$) calcule $\mathrm{Var}([c_0,...,c_d])$ ou les c sont les coordonnées de P dans $B_{d,i}[\frac{c}{2^k},\frac{c+1}{2^k}]$
- addSuccessor($\mathrm{Node}_{k,c}(P)$) consiste simplement à évaluer P en $\frac{c}{2^k}$ pour tester si il s'annule en ce point (auquel cas on a trouvé une racine exacte) et à construire, les noeuds fils de $\mathrm{Node}_{k,c}$(P), c'est à dire essentiellement calculer $k+1$, $2\ c$, $2\ c+1$ et exprimer P dans les bases $B_{d,i}[\frac{2\,c}{2^{k+1}},\frac{2\,c+1}{2^{k+1}}]$ et $B_{d,i}[\frac{2\,c+1}{2^{k+1}},\frac{2\,c+2}{2^{k+1}}]$
- La fonction $<$ est n'importe quel ordre sur $\mathbb{N}^2$.

Comme dans le cas de polynômes exprimés dans la base de monômes, l'utilisation de l'ordre $<_{\mathrm{back}}$ facilite le calcul des noeuds. En effet, étant donnés les coefficients $[b_0,...,b_d]$ de P dans $B_{d,i}[\alpha,\beta]$, l'algorithme de De Casteljau [79] permet de déduire à moindre coût les coefficients de P dans $B_{d,i}[\alpha,\gamma]$ (première liste) et $B_{d,i}[\gamma,\beta]$ (deuxième liste) :

Algorithme `DeCasteljau(`P`,`α`,`β`,`γ`)`

$u:=\frac{\beta-\gamma}{\beta-\alpha}$; $v:=\frac{\gamma-\alpha}{\beta-\alpha}$;

$b_j^{(0)}=b_j,\ j=0...d$

`For i=1..d`

  `for j=0..d-i`

    $b_j^{(i)}=u\,b_j^{(i-1)}+v\,b_{j+1}^{(i-1)}$

`Retourner(`$[b_0^{(0)},...,b_0^{(j)},...,b_0^{(d)}]$, $[b_0^{(d)},...,b_j^{(d-j)},...,b_p^{(d)}]$`)`

On peut remarquer que, dans le cas ou $\alpha = \frac{c}{2^k}$, $\beta = \frac{c+1}{2^k}$ et $\gamma = \frac{2\,c+1}{2^{k+1}}$, c'est à dire dans le cas courant pour l'algorithme de bissection, l'algorithme n'effectue que des additions et des multiplications par 2, ce qui le rend d'un coût très voisin du « Taylor shift » utilisé précédemment et permet de faire un parallèle immédiat avec la méthode de Collins/Akritas.

Remarque 2.12. Il est possible de concevoir une transformation relative permettant de calculer les coefficients de P dans $B_{d,i}[\alpha, \beta]$ connaissant les coefficients de P dans $B_{d,i}[\alpha', \beta']$ mais, contrairement au cas des polynômes à coefficients dans la base de monômes, on alourdit alors considérablement l'algorithme en triplant le nombre de calculs (**FR**-[98] algorithme 3.6 et lemme 3.7).

2.6 Complexité - Choix de la variante la plus adaptée

On suppose que P est un polynôme de degré d à coefficients entiers de taille binaire $< L$, et on note $\bar{P}$ sa partie sans facteurs carrés.

Les meilleures bornes de complexité connues sont de la forme $O(N\,A\,S)$ ou N est le nombre de noeuds de l'arbre de calcul, A nombre d'opérations arithmétiques nécessaires pour la visite d'un noeud, S étant le temps requis pour l'opération arithmétique la plus coûteuse. La taille des entiers intervenant dans les calculs, en particulier les scalaires $\frac{c}{2^k}$, est directement fonction de la profondeur de l'arbre de calcul.

Remarque 2.13. On peut supposer la profondeur de l'arbre de calcul identique que l'on considère P ou $\bar{P}$, puisque qu'elle est, dans les deux cas, majorée par $O(\log(\frac{1}{\mathrm{Sep}(P)}))$, qui est uniquement fonction de la distance entre 2 racines. On peut par exemple prendre comme majorant unique $O(d(L+\log(d))$.

Quelle que soit la variante utilisée (Sturm, Descartes, etc.) les résultats de [40] et [39] montrent que N est majoré par $O(d\,(L+\log(d)))$, améliorant sensiblement les bornes que l'on peut obtenir naïvement en multipliant la profondeur maximale de l'arbre $O(d(L+\log(d))$ par sa largeur maximale ($O(d)$) et en particulier celles de [35] et [76].

Remarque 2.14. Comme pour la remarque 2.13, on peut utiliser la borne obtenue en fonction du degré et de la taille des coefficients de P pour majorer le nombre de noeuds de l'arbre de calcul associé à $\bar{P}$ (facile à voir sur la démonstration de ce résultat dans [40]).

Dans le cas de l'utilisation d'une suite de Sturm, le calcul de la suite nécessite $O(d^2)$ opérations arithmétiques, engendrant des coefficients de taille au plus $O(d\ L)$ [59]. A chaque noeud (k, c), cette suite doit être évaluée en $\frac{c}{2^k}$ et $\frac{c+1}{2^k}$, induisant un coût maximum en $O(d^2 M(d(L+\log(d)))$ par noeud, $M(s)$ majorant le coût de la multiplication de deux entiers de taille au plus s. On obtient ainsi un coût total majoré par $O(d^3(L+\log(d))M(d(L+\log(d))))$.

Dans le cas de l'utilisation de la règle de Descartes, on remplace P par $\bar{P}$ ainsi que le coût de l'évaluation de la suite de Sturm par le coût du calcul de $\bar{P}_{k,c}$, qui est de l'ordre du coût d'un « Taylor shift » que ce soit dans le cas de l'utilisation d'une représentation classique ou dans le cas de l'utilisation de bases de Bernstein. La taille maximale d'un coefficient de $\bar{P}_{k,c}$ dans la base de monômes classique se majore facilement en utilisant la formule $\bar{P}_{k,c} = 2^{k\,d}\bar{P}\left(c\left(\frac{X}{\frac{c}{2^k}}+1\right)\right)$. Les changements $X \to X/c$ ou $X \to X/2^k$ induisent une croissance $t \to d\ h + t$, h étant la profondeur maximale de l'arbre, le changement $X \to X+1$ entraîne une croissance $t \to t+d$, ce qui nous donne des coefficients de taille majorée par $O(d^2(L+\log(d))$, que P soit ou non sans facteurs carrés.

Dans le cas d'un polynôme exprimé dans la base de Bernstein, il est facile de remarquer que la croissance des coefficients évolue de façon identique, l'algorithme de De Casteljau se comportant comme un « Taylor shift ». Il faut toutefois prendre quelques précautions supplémentaires dès lors que le polynôme à étudier est initialement exprimé dans la base de monômes usuelle :

Remarque 2.15. Si on suppose P initialement exprimé dans la base de monômes, l'application de l'algorithme **BernsteinBissection** nécessite un changement de base préalable. Ce changement de base induit une croissance de coefficients : si P à des coefficients de taille L dans la base de monômes, ses coefficients dans la base $B_{d,i}[0,1]$ seront alors de taille $O(d\,L)$. Cependant, on peut toujours majorer le nombre de noeuds de l'arbre de calcul ainsi que sa profondeur par $O(d\,(L+\log(d)))$ et remarquer que la croissance due à ce changement de base n'influe pas sur la borne obtenue ci-dessus pour la taille des coefficients des polynômes apparaissant dans la récursion (même raisonnement que pour le remplacement de P par $\bar{P}$).

Ainsi, on obtient un coût global en $O(d\ (L + \log(d))\ T(d^2(L + \log(d)))$ ou $T(s)$ est le coût d'un « Taylor shift » ou de l'application de l'algorithme de De Casteljau (pour $u = 1/2$ et $v = 1/2$) sur un polynôme dont les coefficients sont de taille s. Ceci nous donne une borne en $O(d^5(L + \log(d))^2)$ si on utilise soit l'algorithme de De Casteljau soit la variante simple du Taylor shift présentée plus haut.

Si l'on utilise des opérations asymptotiquement rapides pour les « Taylor shift » [72] dans le cas de polynômes exprimés dans la base de monômes, on obtient alors une borne en $O(d\ (L + \log(d))\, M(d^3\,(L+\log(d)))$.

Pour être raisonnablement complets, il faut considérer le surcoût dû au calcul de $\bar{P}$ connaissant P. Celui-ci nécessite $O(d^2)$ opérations arithmétiques et la taille des coefficients apparaissant en cours de calcul n'excède pas $O(d\ L)$ si un algorithme standard (par exemple de type sous-résultants) est utilisé. On peut donc négliger le coût de ce calcul face au reste.

Le théorème suivant résume les résultats obtenus ci-dessus et lève l'hypothèse « P sans facteurs carrés » présente dans [40] :

Théorème 2.16. *Soit P un polynôme de $\mathbb{Z}[X]$ de degré d dont les coefficients sont dans $]-2^L, 2^L[$.*

*L'algorithme **SturmBissection** à une complexité binaire en $O(d^3(L + \log(d))M(d(L + \log(d))))$, soit en $\tilde{O}(d^{3+\alpha}L^{1+\alpha})$ si l'on suppose que $M(s)$ est en $\tilde{O}(s^\alpha)$*

*Les algorithmes **CollinsAkriatasBissection**, **KrandickBissection**, **RelativeBissection** ont une complexité binaire en $O(d\ (L + \log(d))\ M(d^3\ (L + \log(d)))$ ou encore $O(d^5(L + \log(d))^2)$, cette dernière borne étant également valable pour l'algorithme **BernsteinBissection**.*

Les tests pratiques proposés dans la section 2.8 de ce chapitre confirment la supériorité, en général des méthodes basées sur la règle de Descartes. Aussi, le choix d'un algorithme d'isolation se portera en priorité sur ce type de stratégie. Parmi les algorithmes utilisant une représentation de P dans la base de monômes, il est clair que la préférence ira sans discussion à l'algorithme **RelativeBissection** puisqu'il est optimisé pour l'occupation en mémoire et présente une rapidité d'exécution équivalente aux algorithmes **CollinsAkriatasBissection**, **KrandickBissection**. Cet algorithme sera également préféré à la variante **BernsteinBissection** pour trois raisons :

- même si l'utilisation de l'algorithme de De Casteljau laisse supposer que la variante **BernsteinBissection** effectue moins d'opérations (disons que pour le coût d'un « Taylor shift » on peut visiter simultanément deux noeuds), le calcul en deux étapes ($P_{k,c}$ puis $\mathrm{Var}((1 + X)^d P_{k,c}(\frac{1}{1+X}))$ autorise bon nombre d'optimisations. Par exemple, on peut arrêter le calcul de $(1 + X)^d P_{k,c}(\frac{1}{1+X})$ dès lors que le résultat présente plus de deux variations de signes dans la suite de ses coefficients, rendant cette étape de calcul négligeable pour un nombre conséquent de noeuds;

- l'avantage procuré par l'algorithme de De Casteljau sur le nombre de transformations réalisées est réduit à néant dès lors que l'on désire obtenir un algorithme optimisé en mémoire (Remarque 2.12);
- pour la résolution de polynômes exprimés dans la base de monômes, qui représente le cas qui nous intéresse dans le cadre de ce document, la stratégie **BernsteinBissection** nécessite un changement de base impliquant une croissance significative de la taille des coefficients (Remarque 2.15). Même si cette croissance n'influe que sur la constante des bornes de complexité, elle alourdit considérablement les calculs en pratique.

2.7 Utilisation d'arithmétique en précision fixe

Une façon d'améliorer le temps d'exécution est de limiter la taille des coefficients manipulés. Un moyen direct de le faire est de remplacer l'arithmétique sur les rationnels par une arithmétique flottante en précision fixe. Ceci revient à représenter les nombres « réels » par par des intervalles dont les bornes sont de la forme $m\,2^e$ avec $|m| < 2^p$, p étant la précision de l'arithmétique flottante utilisée. Pour un nombre réel α donné, on notera $\lfloor \alpha \rceil_p$ un intervalle à bornes rationnelles de précision p contenant α et par $|\lfloor \alpha \rceil_p|$ sa longueur, c'est à dire la distance entre borne de droite et sa borne de gauche, représentant l'erreur maximale effectuée si l'on approche la valeur de α par un point de $\lfloor \alpha \rceil_p$.

On peut facilement spécifier une arithmétique travaillant avec ce type de représentation et encodant les 4 opérations arithmétiques de base. Nous utiliserons ici le même standard que dans [110], [108] ou encore [109], à savoir :

- $\lfloor \alpha \rceil_p$ peut avoir comme borne $+\infty$ ou $-\infty$
- $\lfloor \alpha \rceil_p \,\mathrm{op}\, \lfloor \beta \rceil_p = \lfloor \gamma \rceil_p \supseteq \lfloor \alpha' \,\mathrm{op}\, \beta' \rceil_p, \forall \alpha' \in \lfloor \alpha \rceil_p, \forall \beta' \in \lfloor \beta \rceil_p$ pour $\mathrm{op} = +, -, \star, /$
- $\mathrm{sign}(\lfloor \alpha \rceil_p) = 0$ ssi les deux bornes de $\lfloor \alpha \rceil_p$ sont nulles
- $\mathrm{sign}(\lfloor \alpha \rceil_p) = +$ (resp. -) ssi les deux bornes de $\lfloor \alpha \rceil_p$ sont strictement positives (resp. négatives)
- $\mathrm{sign}(\lfloor \alpha \rceil_p) = ?$ ssi les deux bornes de $\lfloor \alpha \rceil_p$ sont de signes différents.

Remarque 2.17. Un inconvénient majeur de ce type d'encodage est que l'erreur d'approximation augmente avec le nombre d'opérations : $\lfloor \alpha \rceil_p \,\mathrm{op}\, \lfloor \beta \rceil_p = \lfloor \gamma \rceil_p \supseteq \lfloor \alpha' \,\mathrm{op}\, \beta' \rceil_p, \forall \alpha' \in \lfloor \alpha \rceil_p, \forall \beta' \in \lfloor \beta \rceil_p$ pour $\mathrm{op} = +, -, \star, /$, cette inclusion étant généralement stricte.

L'idée d'utiliser ce type d'arithmétique pour l'encodage des scalaires apparaissant dans les algorithmes d'isolation présentés dans les sections précédentes est naturelle. En effet, les calculs de bornes sur le nombre de racines dans un intervalle donné ne requièrent que la connaissance du signe d'expressions et laissent donc, a priori, une bonne marge de manoeuvre sur la précision des approximations que l'on pourrait utiliser.

Seule l'apparition d'un zéro dans des intervalles ferait échouer le procédé. Il ne peut y avoir que deux causes possibles à un tel phénomène

- une cause intrinsèque : représenter un polynôme à coefficients rationnels ou réels $P = \sum_{i=0}^{d} a_i X^i$ par un polynôme dont les coefficients sont des intervalles $\lfloor P \rceil_p = \sum_{i=0}^{d} \lfloor a_i \rceil_p X^i$ revient à considérer tous les polynômes de la forme $Q = \sum_{i=0}^{d} b_i X^i$ avec $b_i \in \lfloor a_i \rceil_p$. En particulier, ces polynômes n'ont pas forcément tous le même nombre de racines et les signes des expressions à calculer pour compter ou borner leur nombre ne sont donc pas forcément constants;
- les calculs itératifs : au regard de la remarque 2.17, l'accumulation d'opérations arithmétiques augmente mécaniquement l'erreur d'approximation des scalaires par des intervalles.

Sur le deuxième point, les méthodes d'isolation présentées dans les sections précédentes ne se comportent pas de façon similaires. Pour les méthodes basées sur la règle de Descartes, l'accumulation d'erreurs est essentiellement due au calcul itératif des polynômes $P_{k,c}$: l'erreur sera fonction du degré de P mais surtout de la valeur maximale de k (profondeur de l'arbre de calcul), c'est à dire de la localisation des racines. Pour les méthodes basées sur un calcul de suite de Sturm, le calcul ne dépend pas directement de la localisation des racines et est connu pour être numériquement très instable, ce qui se traduit, en termes d'arithmétiques d'intervalles, par une augmentation rapide de la largeur des intervalles, rendant ce type de stratégie très rapidement inutilisable, y compris pour des degrés et tailles de coefficients relativement faibles. La suite de cette étude ne considérera donc que les algorithmes utilisant la règle de Descartes.

L'idée d'utiliser une arithmétique d'intervalles n'est pas nouvelle puisqu'on la retrouve dans [73] ou [29], mais ce type de stratégie semblait poser quelques problèmes de terminaison. La preuve de terminaison systématique est faite dans **FR**-[121] modulo une modification des spécifications de l'algorithme; Au lieu de retourner exclusivement une liste d'intervalles d'isolation, la fonction de base INTERVALBISSECTION décrite ci-dessous retourne deux listes :

- une liste $\mathrm{Isol}(P)$ contenant des intervalles d'isolation, c'est à dire des intervalles pour lesquels la règle des signes à pu être calculée sans ambiguïté (par exemple on peut imaginer qu'aucun intervalle calculé ne contienne 0)
- une liste $\mathrm{Error}(P)$ contenant des intervalles pour lesquels le calcul du nombre de changements de signes nécessaire à l'application de la règle de Descartes est ambigu (on peut imaginer que certains intervalles sont de signes indéfinis induisant une borne pouvant prendre une valeur >1 ou l'une des valeurs 0 ou 1).

On schématise la gestion du caractère incertain de la borne en modifiant les spécifications de la fonction ROOTBOUND. Plus précisément, on suppose maintenant que cette fonction peut retourner la valeur ? dès lors qu'il est impossible de décider si la borne obtenue pour un noeud (k,c) est $0, 1$ ou >1.

Algorithme `IntervalBissection`$(P, \mathrm{getNode}, <, \mathrm{addSuccessor}, \mathrm{RootBound})$
$\mathrm{Exact}(P) \leftarrow \emptyset$, $\mathrm{Isol}(P) \leftarrow \emptyset$, $\mathrm{Error} = \emptyset$ $T \leftarrow \mathrm{Node}_{0,0}(P)$
`While` $T \neq \emptyset$ `do`
  $\mathrm{Node}_{k,c}(P) \leftarrow \mathrm{getNode}(T, <)$
  $s \leftarrow \mathrm{RootBound}(\mathrm{Node}_{k,c}(P))$
  `if` $s = ?$ `then` $\mathrm{Error}(P) = \mathrm{Error}(P) \cup \{(k,c)\}$
  `else`
    `If` $s = 1$ `then` $\mathrm{Isol}(P) \leftarrow \mathrm{Isol}(P) \cup \{(k,c)\}$
    `If` $s > 1$ `then` $(T, \mathrm{Exact}(P)) \leftarrow \mathrm{addSuccessors}(\mathrm{Node}_{k,c}(P))$
`Return`$(\mathrm{Isol}(P), \mathrm{Exact}(P), \mathrm{Error}(P))$

Il est clair que cet algorithme terminera toujours, par contre, on n'obtiendra pas forcément un résultat complet ($\mathrm{Error}(P) \neq \emptyset$).

Dès lors que P est connu exactement (coefficients rationnels) et qu'il est sans facteurs carrés, l'unique cause d'échec ($\mathrm{Error}(P) \neq \emptyset$) est l'accumulation d'erreurs d'approximations (élargissement des intervalles représentant les scalaires). Comme suggéré dans, **FR**-[121], on peut alors, soit repartir en exact, soit simplement augmenter la précision de l'arithmétique et étudier les racines de P dans chaque intervalle de la liste $\mathrm{Error}(P)$. Notons qu'à défaut de trouver le résultat complet, la fonction INTERVALBISSECTION permet au moins de raffiner l'intervalle de recherche des racines.

2.8 Expériences

Les méthodes décrites dans ce chapitre ont toutes été implantées dans le logiciel RS, utilisant donc strictement les mêmes outils et bibliothèques. Ceci permet une mesure de performances relatives, donnée par un ratio par rapport à un autre calcul (voir par exemple le tableau donné en introduction de ce chapitre).

La qualité brute des implantations semble correcte puisqu'au moment ou ces tests ont été réalisés (voir **FR**-[121] et **FR**-[120]), RS surpassait d'autres méthodes/implantations réputées comme les logiciels SACLIB ou encore MPSolve [18].

Dans le tableau ci-dessous, le référentiel des mesures est l'algorithme RelativeBissection. La première colonne est le nom de l'exemple traité, la seconde (Degree) est le degré du polynôme, la colonne Nb. Real indique le nombre de racines réelles. Pour chaque algorithme évalué (**CollinsAkriatasBissection** et **KrandickBissection**) la mesure est constituée de deux nombres séparés par un « / ». Le premier est le ratio du temps de calcul, le deuxième est celui de l'occupation mémoire. Un temps ou une occupation mémoire « ∞ » veut dire supérieur(e) à une limite arbitraire choisie au delà de laquelle le programme a été stoppé. Ces tests sont issus de **FR**-[121] : la limite de temps de calcul était 7200 sec et la limite d'occupation mémoire 256Mb (ce qui induisait souvent une occupation réelle double en pratique).

Name	**Degree**	**Nb. Real**	**CollinsAkritas**	**Krandick**
Chebyshev	900	900	1.2/3.1	1.0/68.7
	1000	1000	1.2/3.5	1.0/92.2
Laguerre	900	900	1.5/18.5	1.4/113.6
	1000	1000	1.3/19.14	?/∞
Wilkinson	900	900	1.3/16.5	1.56/121.1
	1000	1000	1.3/16.7	?/∞
Mignotte	300	4	2.0/88.2	1.0/3.9
	400	4	∞/?	1.0/3.9
P2	1366	50	1.0/14.5	1.0/2.7
Fau2	244	18	1.8/38.7	1.2/7.5
Cyclic 7	924	56	1.0/7.7	1.0/10.5
Katsura 9	512	120	1.0/7	1.2/19.8
Virasoro	256	224	1.8/11.5	3.1/28

L'apport de la variante « mémoire optimale » de la méthode de bissection sur ces quelques exemples classiques est évident, même du point de vue de la rapidité d'exécution. L'algorithme **RelativeBissection** n'a aucune raison d'être plus rapide que les variantes **CollinsAkriatasBissection** et **KrandickBissection** en termes de temps de calcul pur, mais le coût de la gestion de mémoire, en particulier dans l'algorithme **CollinsAkriatasBissection**, peut apparaître de façon sensible. La conclusion que l'on peut tirer est que l'algorithme **RelativeBissection** est incontestablement la meilleure variante.

Le tableau suivant montre sur quelques exemples que l'algorithme hybride est une réelle valeur ajoutée sur des polynômes issus provenant d'applications, son apport étant nettement moins flagrant voir pratiquement nul sur les *benchmarks* connus, sauf sur les polynômes de Mignotte sur lesquels il permet de travailler avec des degrés nettement plus élevés.

Pour ces exemples, la précision utilisée dans la stratégie Interval augmente de 53 à 863 bits, doublant chaque fois qu'il reste des intervalles sur lesquels aucune conclusion n'est possible, puis termine les calculs avec des entiers multi-précision si nécessaire.

On a donc tout intérêt à utiliser systématiquement cette variante, même si l'on repasse rapidement en arithmétique classique. En effet, même si le calcul ne permet pas d'obtenir toutes les racines, il permet en tout cas d'affiner la borne utilisée. Comme pour toute règle il y a dans ces exemples une exception (Virasoro) : cet exemple à la particularité d'avoir un nombre conséquent de racines exactes que l'algorithme classique trouve rapidement et que l'algorithme « Interval » ne peut structurellement pas trouver.

Les conventions utilisées pour le tableau suivant sont les mêmes que pour le test précédent (ici il n'y a pas d'indication sur l'occupation mémoire).

Name	Degree	Nb. Real	INTERVAL
Chebyshev	900	900	0.9
	1000	1000	0.9
Laguerre	900	900	1.0
	1000	1000	0.9
Wilkinson	900	900	1.0
	1000	1000	1.0
Mignotte	300	4	0.06
	400	4	0.05
	600	4	0 ($=\frac{1}{\infty}$)
P2	1366	50	0.2
Fau2	244	18	0.3
Cyclic 7	924	56	0.2
Katsura 9	512	120	0.5
Virasoro	256	224	1.2

Le tableau suivant reprend celui donné en introduction de ce chapitre et montre que l'objectif d'améliorer la boite noire « résolution des polynômes en une variable » a été atteint puisque le poids de ce calcul a été rendu négligeable dans le processus global de résolution des systèmes zéro-dimensionnels alors que cette étape était la plus coûteuse dans la plupart des cas lorsque l'on utilisait les suite de Sturm (en l'occurrence Sturm-Habicht pour ces tests).

On mesure ici les temps de calcul nécessaires pour isoler les zéros réels du premier polynôme d'une Représentation Univariée Rationnelle d'un système. Les conventions sont les mêmes que dans les tests précédents, l'unité de temps étant cette fois le temps de calcul nécessaire pour obtenir une Représentation Univariée Rationnelle (base de Gröbner + RUR).

Exemple	STURM-HABICHT	DESCARTES	Nb. Real	Degree
Fabrice24	3.9	0.002	24	40
camera1s	0.8	0.005	16	20
cpdm5	46.4	0.007	43	163
cyclic6	9.8	0.091	24	156
cyclic7	∞	0.140	56	924
d1	3.6	0.007	16	48
delta2	9.1	0.002	8	40
des18_3	4.3	0.008	6	46
des22_24	1.7	0.009	10	42
ducos7_3	5.7	0.011	168	168
hexa2	8.6	0.002	8	40
hexaglide	13.3	0.015	8	36
i1	∞	0.002	16	66
ilias_k_2	0.7	0.000	14	72
ilias_k_3	0.7	0.000	18	84
ingersoll	8.7	0.002	6	40
katsura6	12.7	0.062	32	64
katsura7	66.0	0.028	44	128
katsura8	∞	0.021	84	256
katsura9	∞	0.026	120	512

Bien que ce n'était pas le but initial, il se trouve que les algorithmes de bissection basés sur la règle de Descartes proposés dans cette partie sont très concurrentiels. Ils dominent largement les méthodes s'appuyant sur une suite de Sturm, les méthodes purement numériques de type Newton ne serait-ce que parce qu'ils calculent assurément toutes les solutions en garantissant la précision et le caractère réel du résultat.

2.9 Extensions

L'isolation des zéros d'un polynôme en une variable n'est qu'une des multiples fonctionnalités nécessaires pour bon nombre d'algorithmes présentés dans ce document. Il faut de plus avoir un procédé de raffinement permettant d'obtenir des approximations numériques certifiées, mais une fonction qui sera essentielle par la suite est l'évaluation d'un polynômes en les zéros d'un autre, soit simplement pour en connaître le signe, soit pour avoir en une valeur approchée certifiée.

Dans [13], le signe des éléments d'une liste de polynômes Q_1, ..., Q_s en les zéros d'un polynôme P est calculé par un algorithme évitant l'isolation des zéros de P (« simultaneous inequalities »), reprenant les idées présentes dans [17].

Si on suppose, pour simplifier, que tous les polynômes sont de degré d et ont des coefficients entiers de taille t la complexité de l'algorithme est alors celle de $O(s\ p)$ calculs de suites de Sturm généralisées avec des polynômes de degrés au plus $O(d\log(d))$ dont les coefficients sont de taille au plus $O((d\log(d))(\ t+\log(d\log(d)))$. Globalement, on peut voir que ce coût est équivalent à $O(s)$ calculs d'isolation de racines de polynômes de degrés d et de coefficients entiers de taille t.

Malheureusement, cette méthode ne permet aucunement de calculer des approximations numériques des valeurs des Q_i en les zéros de P.

Dans cette section, nous montrons que les méthodes d'isolation basées sur la bissection et la règle de Descartes permettent de remplacer l'algorithme « simultaneous inequalities » par un algorithme de complexité équivalente mais donnant, de plus, une approximation numérique certifiée des valeurs des Q_i en les racines de P.

Soient P et Q deux polynômes sans facteurs carrés. Connaissant les intervalles d'isolation de P et Q et supposant qu'ils n'aient pas de racines communes, il suffit de raffiner les intervalles d'isolation de ces 2 polynômes jusqu'à ce qu'ils soient de largeur inférieure à la distance minimale entre une racine de P et une racine de Q pour conclure : une simple évaluation de Q en n'importe quel point d'un intervalle d'isolation de P suffit alors. Il n'est bien sûr pas question, en pratique, de calculer une borne sur la distance minimale entre les racines de Q et celles de P, mais cet argument assure la terminaison de la méthode suivante :

- Soient $G=\gcd(P,Q)$ et $P_1=P/G$, $Q_1=Q/G$, $(a_i,b_i), i=1...n_P$ (resp. $(a'_i,b'_i), i=1...n_Q$) les intervalles d'isolation de P (resp. Q). Par convention, on suppose que si (a_i,b_i) est un intervalle d'isolation d'un polynôme P, si $P(a_i)=0$ ou $P(b_i)=0$ alors $a_i=b_i$

- On calcule les intervalles d'isolation (a''_i,b''_i) et (a''_i,a''_i) de P correspondant aux points ou Q s'annule en évaluant G aux bornes des intervalles $(a_i,\ b_i)$ ou $(a_i,\ b_i)$, puis on les enlève des listes (a_i,b_i) et (a'_i,b'_i) :$\{(a_i,b_i), i=1...n_{P_1}\}$ est désormais la liste des intervalles d'isolation de P_1, $\{(a'_i,b'_i), i=1...n_{Q_1}\}$ celle de Q_1.

- Pour tout couple d'intervalles $(a_i,b_i),(a'_j,b'_j)$:

  - [*] si $(a_i,\ b_i)\cap(a'_j,\ b'_j)=\emptyset$, Q_1 (et donc Q) est de signe constant au dessus de $(a_i,\ b_i)$;

  - sinon, on définit $(c_i,d_i)=(a_i,b_i)\cap(a'_j,b'_j)$

    - Si $(c_i,,\ d_i)=(a_i,\ b_i)$, ou $(c_i,,\ d_i)=(a'_i,\ b'_i)$, on raffine $(a_i,\ b_i)$ et $(a'_i,\ b'_i)$ et on reprend à l'étape [*]

    - Si $\text{sign}(Q_1(c_i))=0$ ou $\text{sign}(Q_1(d_i))=0$ ou $\text{sign}(P_1(c_i))=0$ ou $\text{sign}(P_1(d_i))=0$ on remplace $(a'_i,\ b'_i)$ et/ou $(a_i,\ b_i)$ par $(c_i,\ c_i)$ et/ou $(d_i,\ d_i)$ et on reprend à l'étape [*]

    - Si $\text{sign}(Q_1(c_i))\neq\text{sign}(Q_1(d_i))$ et $\text{sign}(P_1(c_i))\neq\text{sign}(P_1(d_i))$, on remplace $[a'_i,\ b'_i]$ et $[a_i,b'_i]$ par $[c_i,d_i]$ et on reprend à l'étape [*]

Connaissant les intervalles d'isolation de P et Q, la complexité de ce processus dépend essentiellement de la distance entre les racines de P_1 et de celles de Q_1. En termes de complexité, cette distance est minorée par $\mathrm{Sep}(PQ)$. Supposons que P (resp. Q) sont des polynômes de degrés au plus d et que leurs coefficients soient de taille au plus t. Comme PQ à des coefficients de taille $O(t+d)$ on peut donc voir que $\mathrm{Sep}(P\,Q)$ est asymptotiquement peu différente de $\mathrm{Sep}(P)$ ou $\mathrm{Sep}(Q)$, ce procédé n'est donc, en théorie, pas plus coûteux que l'isolation des zéros des polynômes P et Q, le surcoût étant essentiellement le raffinement des intervalles d'isolation de $\mathrm{Sep}(P)$ et $\mathrm{Sep}(Q)$ pour passer d'intervalles de largeur $O(\mathrm{Sep}(P))$ (resp. $\mathrm{Sep}(Q)$) à des intervalles de largeur $O(\mathrm{Sep}(PQ))$, soit de l'ordre de $O(d^2)$ évaluations de P (ou Q) en des points de la forme $\frac{c}{2^k}$ avec $k = O(d\,(t+d+\log(d)))$ et $c < 2^k$.

Théorème 2.18. *Soient P, Q_1, ..., Q_s des polynômes de degrés au plus d et ayant des coefficients entiers de taille au plus t. Isoler les racines réelles $\{\alpha_1, ..., \alpha_l\}$ de P par des intervalles $(a_i, b_i)_{i=1...l}$ et calculer des intervalles $(a_i^{(j)}, b_i^{(j)})_{i=1...l}^{j=1...s}$ tels que $Q_j((a_i, b_i)) \subset (a_i^{(j)}, b_i^{(j)})$ et que Q_j soit de signe constant au dessus de (a_i, b_i), nécessite au plus $O(s\ \mathrm{Isol}(d, t))$, $\mathrm{Isol}(d, t)$ étant le coût de l'isolation des racines d'un polynôme de degré d avec des coefficients entiers de taille au plus t par l'algorithme RelativeBissection.*

Remarque 2.19. Le théorème précédent utilise une même borne pour le degré de tous les polynômes ainsi que pour la taille de leurs coefficients : c'est la situation qui nous intéresse le plus au regard de l'utilisation de ce type de fonction dans le cadre de la résolution de systèmes zéro-dimensionnels. On peut facilement prendre en compte séparément les degrés des Q_i ainsi que la taille de leurs coefficients mais la borne finale, d'expression plus complexe, n'apporte pas d'amélioration.

Enfin, il est clair que l'on peut étendre facilement ce type de méthode à la comparaison de deux nombres algébriques réels lorsque chacun d'entre eux est défini par un polynôme et un intervalle d'isolation et à bien d'autres opérations sur les nombres algébriques réels, mais nous sortirions du cadre de ce document.

2.10 Conclusion

Le résultat principal de ce chapitre est un algorithme dont les spécifications sont les suivantes :

SOLVE-UNIVARIATE

- **Input** : $P \in \mathbb{Q}[T]$, $\mathcal{F} = \{q_1, ..., q_l\}, q_i \in \mathbb{Q}[T]$, p and integer
- **Output** :
  - FAIL if $P = 0$
  - $\{(a_i^j, b_{i,k}^j), i = 1...d, j = 0...l\}$ with $a_i^j, b_i^j \in \mathbb{Q}$, such that
    - $(a_i^j,\ b_i^j) = [a_i^j,\ a_i^j] = \{a_i^j\}$ if $a_i^j = b_i^j$ and $(a_i^j,\ b_i^j) =]a_i^j,\ b_i^j]$ if $a_i^j \neq b_i^j$. Moreover, a_i^j and b_i^j have the same sign;
    - $\alpha \in \boldsymbol{V}(P) \cap \mathbb{R}^n$ iff $\exists i, \alpha \in (a_i^0, b_i^0)$;
    - (a_i^0, b_i^0) contains a unique element of $\boldsymbol{V}(P) \cap \mathbb{R}^n$
    - if $\alpha \in \boldsymbol{V}(P) \cap \mathbb{R}^n$ then $\exists i, \alpha \in (a_i^0, b_i^0)$ and $q_j(\alpha) \in (a_i^j, b_i^j), \forall j = 1...l$
    - $|a_i^j - b_i^j| < 1/2^p, \forall i = 1...d, j = 0...l$.
  - a function REFINE taking as input P, $\mathcal{F}$, $\{(a_i^j,\ b_i^j),\ i = 1...d,\ j = 0...l\}$ and an integer $p' > p$, which returns $\{(c_i^j, d_i^j), i = 1...d, j = 0...l\}$ with the same properties as $\{(a_i^j, b_i^j), i = 1...d, j = 0...l\}$ but such that $|c_i^j - d_i^j| < 1/2^{p'}, \forall i = 1...d, j = 0...l$.

L'objectif initial, qui était de diminuer la part du traitement des polynômes en une variable à coefficients rationnels dans le processus global de résolution des systèmes zéro-dimensionnels proposé dans ce document, est atteint. Pour la suite, on retiendra de ce chapitre l'existence de boites noires efficaces permettant de traiter les problèmes en une variable énumérés en introduction de ce document : comptage, isolation, évaluation de polynômes en les racines d'un autre. Le terme « efficace » est a priori relatif au reste des calculs nécessaires, mais l'expérience montre que l'algorithme **RelativeBissection** dans sa version « arithmétique par intervalles » est efficace de façon absolue.

Il est donc désormais utilisé en dehors du cadre initialement fixé, en particulier pour tracer des courbes planes données sous forme implicite. Cette application quasi-directe a permis de résoudre « graphiquement mais de façon certifiée » plusieurs problèmes, notamment certains systèmes dépendant de paramètres, mais l'application la plus significative est celle liée à l'étude des ridges en géométrie algorithmique, présentée dans le chapitre 6.

Bien que l'algorithme **RelativeBissection** soit généralisable au cas de polynômes à coefficients dans un corps archimédien, il ne peut pas se substituer aux solutions connues pour la résolution de polynômes en une variable à coefficients dans certains autres corps. Par exemple, pour des polynômes à coefficients dans le corps des séries de Puiseux, il faudra se retourner vers l'algorithme de Newton-Puiseux si l'on désire calculer les premier termes du développement des solutions ou simplement à Sturm-Habicht si il ne s'agit que de les compter.

Chapitre 3
Systèmes zéro-dimensionnels

3.1 Introduction

Lorsque l'on pense « résolution de systèmes algébriques » en calcul scientifique, on sous-entend généralement « système admettant un nombre fini de solutions complexes ». Ce chapitre est donc un passage important de ce document autant pour l'étude globale que pour son intérêt dans l'absolu.

En « calcul scientifique », le problème est jugé difficile à aborder, au point d'être, en général, soit évité par des linéarisations ou des reformulations, soit résolu essentiellement localement, c'est à dire en ayant déjà une idée de l'emplacement de certaines solutions, avec un faible niveau de garantie sur le résultat (on se contente en général de vérifier que la substitution des racines trouvées dans le système initial donne « presque » zéro).

Il y a donc énormément de problèmes ouverts ou la résolution certifiée d'un système zéro-dimensionnel est essentielle. L'un de ces challenges nous a servi d'application « fil rouge » pour bon nombre d'études et en particulier pour celle présentée ici : la résolution du modèle géométrique direct des robots parallèles, décrite dans le chapitre 6. La formulation de ce problème est « naturellement » algébrique et certains résultats importants sur le sujet (par exemple montrer que le problème admet au plus 40 solutions complexes [81],[95],[106]) ont été obtenus par des techniques de calcul algébrique. Dans le chapitre 6, d'autres applications ayant servi à la mise au point des algorithmes présentés dans ce chapitre sont décrites. On peut citer, par exemple, les calculs de topologie de ridges qui ont permis de régler les différents algorithmes pour obtenir une solution par défaut globalement satisfaisante.

Outre le fait que la résolution des systèmes zéro-dimensionnels est une fin en soi, ce problème est central dans le travail présenté dans ce document puisque qu'il constitue un prérequis (boite noire) pour tous les autres algorithmes (systèmes généraux, systèmes paramétrés).

Pour ne pas compliquer inutilement la rédaction, les systèmes étudiés seront supposés par défaut à coefficients rationnels, la plupart des résultats présentés se généralisant facilement à d'autres corps. Les notations suivantes seront systématiquement utilisées :

Notation 3.1. *Soient $p_1, ..., p_s$ des polynômes de $\mathbb{Q}[X_1, ..., X_n]$.*

- *$\mathcal{S}$ est le système d'équations $\{p_1 = 0, ..., p_s = 0\}$*
- *$\mathcal{I} = \langle p_1, ..., p_s \rangle$ est l'idéal de $\mathbb{Q}[X]$ engendré par les polynômes définissant $\mathcal{S}$.*
- *$\boldsymbol{V}(\mathcal{I}) = \{x \in \mathbb{C}^n, p_1(x) = 0, ..., p_s(x) = 0\}$*

3.2 Des systèmes zéro-dimensionnels à l'algèbre linéaire

A la différence des systèmes généraux, beaucoup de problèmes liés à la résolution des systèmes zéro-dimensionnels se traduisent naturellement en problèmes d'algèbre linéaire classiques. Les deux résultats qui suivent établissent l'essentiel du parallèle :

Théorème 3.2. *(Voir [13]) $\frac{\mathbb{C}[X_1, ..., X_n]}{\mathcal{I}}$ est un $\mathbb{C}$-espace vectoriel de dimension finie, produit fini des localisés de $\frac{\mathbb{C}[X_1, ..., X_n]}{\mathcal{I}}$ en les points de $\boldsymbol{V}(\mathcal{I})$: $\frac{\mathbb{C}[X_1, ..., X_n]}{\mathcal{I}} = \prod_{\alpha \in \boldsymbol{V}(\mathcal{I})} \left(\frac{\mathbb{C}[X_1, ..., X_n]}{\mathcal{I}}\right)_\alpha$. Les anneaux locaux $\left(\frac{\mathbb{C}[X_1, ..., X_n]}{\mathcal{I}}\right)_\alpha$ sont eux mêmes des $\mathbb{C}$-espaces vectoriels de dimension finie, leur dimension $\mu(\alpha)$ définit la multiplicité de α, la dimension de $\frac{\mathbb{C}[X_1, ..., X_n]}{\mathcal{I}}$ est donc le nombre de zéros de $\mathcal{I}$ comptés avec multiplicités.*

Une relation simple entre l'algèbre quotient $\frac{\mathbb{C}[X_1,...,X_n]}{\mathcal{I}}$ et les valeurs des coordonnées des solutions est donnée par :

Théorème 3.3. *(Voir [13]) Soit $h \in \mathbb{C}[X_1, ..., X_n]$, on définit : $m_h : \begin{array}{ccc} \frac{\mathbb{C}[X_1,...,X_n]}{\mathcal{I}} & \longrightarrow & \frac{\mathbb{C}[X_1,...,X_n]}{\mathcal{I}} \\ \bar{u} & \mapsto & \overline{h\,u} \end{array}$, l'endomorphisme de multiplication par h dans $\frac{\mathbb{C}[X_1,...,X_n]}{\mathcal{I}}$. Les valeurs propres de m_h sont exactement les $h(\alpha)$ avec multiplicité $\mu(\alpha)$.*

Connaître une base de $\frac{\mathbb{C}[X_1,...,X_n]}{\mathcal{I}}$ et savoir calculer les matrices des endomorphismes m_{X_i} dans cette base, permet donc, en théorie, de calculer toutes les valeurs possibles de toutes les coordonnées des points de $\boldsymbol{V}(\mathcal{I})$.

Si l'on suppose que $\mathcal{I} \subset \mathbb{Q}[X_1, ..., X_n]$, il existe plusieurs façons de calculer de façon exacte ces objets (notons au passage que $\frac{\mathbb{C}[X_1,...,X_n]}{\mathcal{I}} = \mathbb{C} \otimes \frac{\mathbb{Q}[X_1,...,X_n]}{\mathcal{I}}$ et que l'on peut alors supposer les m_{X_i} à coefficients rationnels). La plus connue est une application directe *des bases de Gröbner* (voir [34] ou [16] pour une introduction), mais on peut citer également les résultat de [136] qui fournissent exactement la même information. Sans rentrer plus dans les détails, nous supposerons que ce qui suit représente le prérequis pour les résultats présentés dans ce chapitre :

Notation 3.4. *Pour tout idéal $\mathcal{I} \subset \mathbb{Q}[X_1, ..., X_n]$*

- *on sait tester si $\mathcal{I}$ est zéro-dimensionnel ($\#\boldsymbol{V}(\mathcal{I}) < \infty$)*
- *on sait calculer une base de monômes $\mathcal{B} = \{w_1, ..., w_D\}$ de $\frac{\mathbb{Q}[X_1,...,X_n]}{\mathcal{I}}$ telle que $w_1 = 1$ et , $\forall j = 1...D, \exists k\, w_j = X_k w_i$*
- *on sait calculer les matrices (à coefficients rationnels) des applications m_{X_i} dans $\mathcal{B}$.*

Supposant connue une base de $\frac{\mathbb{C}[X_1,...,X_n]}{\mathcal{I}}$ ainsi que les matrices m_{X_i} dans cette base, il est alors possible de calculer toutes les valeurs de toutes les coordonnées des points de $\boldsymbol{V}(\mathcal{I})$, simplement en calculant les valeurs propres des m_{X_i}. Dans le cas réel, ceci peut par exemple se faire de façon certifiée, en isolant les racines des polynômes caractéristiques des m_{X_i} par l'algorithme du chapitre 2 ou encore par des techniques numériques [8],[97], la difficulté principale étant de savoir les combiner pour obtenir les points recherchés ...

Si on est chanceux, une base de $\frac{\mathbb{Q}[X_1,...,X_n]}{\mathcal{I}}$ sera par exemple constituée exclusivement de puissances de X_1. En principe, le système se réécrit alors sous la forme $f_1(X_1) = 0$, $X_2 - f_2(X_1) = 0, ..., X_n - f_n(X_1) = 0$, ou $f_1(X_1)$ est le polynôme minimal de m_{X_1}. En dehors de toute considération pratique, ceci peut permettre de conclure en calculant les valeurs propres de m_{X_1} puis en les reportant dans les expressions $X_i - f_i(X_1)$.

Définition 3.5. *Lorsque $\frac{\mathbb{Q}[X_1,...,X_n]}{\mathcal{I}}$ est engendré par des puissances pures de X_i, on dit que l'idéal $\mathcal{I}$ est* shape position *pour X_i.*

Tous les idéaux ne sont bien sur pas *shape position* [15] pour une variable, un contre exemple étant donné par le système $\{(X_1 - 1)\,(X_1 - 2), (X_2 - 1)\,(X_2 - 2)\}$. Sur cet exemple, on ne peut d'ailleurs pas échanger les rôles des variables X_1 et X_2 dans l'espoir de se ramener à un cas favorable. Ce problème est partiellement contourné ([21],[80],[107],[55],[61],[54],[4],[56], etc.) par l'introduction de changements de variables et le calcul de ce que nous nommerons par la suite un *élément séparant* $\boldsymbol{V}(\mathcal{I})$.

Définition 3.6. *Soit $h \in \mathbb{Q}[X_1, ..., X_n]$. On dit que h sépare $\boldsymbol{V}(\mathcal{I})$ si $x \longrightarrow h(x)$ est injective sur $\boldsymbol{V}(\mathcal{I})$.*

Si l'algèbre $\frac{\mathbb{Q}[X_1,...,X_n]}{\mathcal{I}}$ est cyclique, on peut, par définition, toujours trouver un polynôme h tel que $\frac{\mathbb{Q}[X_1,...,X_n]}{\mathcal{I}}$ soit engendré par des puissances de $\bar{h}$, ce qui est équivalent, dans ce cas, à trouver un élément h séparant $\boldsymbol{V}(\mathcal{I})$. Le système se réécrit alors sous la forme $f(T)$, $X_1 = f_1(T), ..., X_n = f_n(T)$, généralisant la stratégie exposée ci-dessus. Mais encore une fois, ceci n'est pas le cas général, puisque, par exemple, le système $\{X_1^2, X_1 X_2, X_2^2\}$ ne pourra jamais se réécrire sous cette forme **FR**-[114].

Afin d'éviter de nombreuses paraphrases par la suite, on identifie une sous-classe de ces systèmes particuliers :

Définition 3.7. *Lorsque* $\frac{\mathbb{Q}[X_1,...,X_n]}{\mathcal{I}}$ *est engendré par des puissances pures d'une combinaison linéaire des variables* $X_1,...,X_n$, *on dira que l'idéal* $\mathcal{I}$ *est* shape position.

Autant trouver une base de $\frac{\mathbb{Q}[X_1,...,X_n]}{\mathcal{I}}$ constituée exclusivement de puissances d'un de ses éléments est structurellement impossible dans le cas général, autant trouver un élément séparant $\boldsymbol{V}(\mathcal{I})$ est toujours possible (d'après le théorème des zéros de Hilbert, à trouver un élément primitif de $\frac{\mathbb{Q}[X_1,...,X_n]}{\sqrt{\mathcal{I}}}$). Presque toutes les formes linéaires séparent un ensemble donné, et on peut même restreindre la recherche :

Lemme 3.8. *(Voir [13]) Supposant que* $\#\boldsymbol{V}(\mathcal{I}) = d$, *la famille* $\{X_1 + i\,X_2 + ... + i^{n-1}X_n, i = 1...n\frac{d\,(d-1)}{1}\}$ *contient au moins un élément séparant* $\boldsymbol{V}(\mathcal{I})$.

Beaucoup de méthodes de résolution utilisent cette notion d'élément séparant, à commencer par certaines méthodes numériques permettant une diagonalisation simultanée des matrices m_{X_1} et donc de combiner les coordonnées des solutions pour obtenir les points recherchés (par exemple dans [8]). Celles-ci n'offrent toutefois aucune garantie quant à la précision du résultat et encore moins sur le caractère réel des solutions, elles sortent donc du cadre de l'étude proposée dans ce document. Dans la section qui suit, on cherchera un autre biais pour calculer des paramétrisations formelles telles que celles que l'on peut facilement obtenir dans le cas d'idéaux *shape position*, mais nous terminons cette section avec quelques applications relativement directes des résultats introduits ci-dessus.

Le lemme 3.8 fournit un algorithme simple permettant de calculer un élément séparant :

Pour $i = 1...n\,\frac{d\,(d-1)}{1}$, calculer la matrice de $m_{X_1+iX_2+...+i^{n-1}X_n} = m_{X_1} + i\,m_{X_2} + ... + i^{n-1}m_{X_n}$ dans la base $\mathcal{B}$ (simple combinaison linéaire des matrices m_{X_i} supposées connues d'après la Notation 3.4) et calculer d_i , le degré de la partie sans facteurs carrés de son polynôme caractéristique. Le éléments séparants $\boldsymbol{V}(\mathcal{I})$ contenus dans $\{X_1 + i\,X_2 + ... + i^{n-1}X_n, i = 1...n\frac{d\,(d-1)}{1}\}$ sont ceux pour lesquels d_i est maximal.

Une conséquence directe est que l'on peut alors calculer explicitement le nombre de solutions distinctes réelles ou complexes de n'importe quel système zéro-dimensionnel :

Lemme 3.9. *Si* t *sépare* $\boldsymbol{V}(\mathcal{I})$, *et si* polchar$(m_t)$ *désigne le polynôme caractéristique de* m_t, *alors* $\#\boldsymbol{V}(\mathcal{I}) = \#\boldsymbol{V}(\text{polchar}(m_t))$. *Dès lors que* $t \in \mathbb{Q}[X_1,...,X_n]$, *on a de plus* $\#(\boldsymbol{V}(\mathcal{I}) \cap \mathbb{R}^n) = \#(\boldsymbol{V}(\text{polchar}(m_t)) \cap \mathbb{R}^n)$.

Bien que « presque toute » forme linéaire soit séparante, on exclura les choix « au hasard » au regard des contraintes fortes énoncées en introduction de ce document. La conjonction des deux lemmes précédents donne un algorithme permettant de construire explicitement une forme linéaire assurément séparante. Un tel « algorithme » est extrêmement coûteux puisqu'il nécessite le calcul de $n\,\frac{d\,(d-1)}{1}$ polynômes caractéristiques, mais il montre au moins que de tels éléments peuvent être explicitement calculés. Pour le calcul du nombre de solutions d'un système, on lui préférera (étude de complexité dans la section 3.4), la méthode suggérée par :

Théorème 3.10. *([104],[16],[105]) Pour tout* $h \in \mathbb{Q}[X_1,...,X_n]$, *on définit :*

$$\begin{array}{rccc} q_h\colon & \frac{\mathbb{Q}[X_1,...,X_n]}{\mathcal{I}} & \longrightarrow & \mathbb{Q} \\ & f & \mapsto & \text{Trace}(f^2\,h) \end{array}$$

Trace(P) *désignant la trace de* m_P *pour tout* $P \in \mathbb{Q}[X_1,...,X_n]$
Alors :

- $\text{rank}(q_h) = \#\{x \in \boldsymbol{V}(\mathcal{I}), h(x) \neq 0\}$
- $\text{signature}(q_h) = \#\{x \in \boldsymbol{V}(\mathcal{I}) \cap \mathbb{R}^n, h(x) > 0\} - \#\{x \in \boldsymbol{V}(\mathcal{I}) \cap \mathbb{R}^n, h(x) < 0\}$

En s'appuyant sur le théorème 3.10 dans le cas ou $h = 1$ et connaissant les prérequis que nous nous sommes donnés (Notation 3.4) on obtient simplement un algorithme permettant de compter le nombre de points de $\boldsymbol{V}(\mathcal{I})$ et de $\boldsymbol{V}(\mathcal{I}) \cap \mathbb{R}^n$. En effet, connaissant les matrices m_{X_i} dans une base $\mathcal{B} = \{w_1, ..., w_D\}$ de $\frac{\mathbb{Q}[X_1, ..., X_n]}{\mathcal{I}}$ on peut calculer $m_{w_i w_j \bar{h}}$ et en déduire la matrice de q_h dans $\mathcal{B}$.

On peut remarquer que le théorème 3.10 donne également une généralisation des résultats sur les suites de Sturm (généralisées) au cas des systèmes zéro-dimensionnels. On peut donc, au moins dans le principe, ainsi compter le nombre de racines distinctes d'un système zéro-dimensionnel mais également connaître le signe de polynômes en ces racines (algorithme « simultaneous inequalities » proposée dans [122] et repris dans [13]).

3.3 Résolution Symbolique : La Représentation Univariée Rationnelle (RUR)

De la section précédente, on peut retenir que modulo le choix d'un élément séparant $t \in \mathbb{Q}[X_1, ..., X_n]$, on peut réécrire facilement le système, **dans certains cas**, sous la forme $f_t(T) = 0$, $X_1 - f_{1,t}(T) = 0$, ..., $X_n = f_{n,t}(T) = 0$ et définir ainsi un isomorphisme d'ensembles algébriques :

$$\begin{array}{rccc} \phi_t\text{:} & \boldsymbol{V}(\mathcal{I}) & \longrightarrow & \boldsymbol{V}(f_t) \\ & \alpha = (\alpha_1, ..., \alpha_n) & \mapsto & t(\alpha) \\ & (f_{t,X_n}(\beta), ..., f_{t,X_n}(\beta)) & \leftarrow\!\shortmid & \beta \end{array}$$

Ce type de représentation des solutions rejoint la philosophie de Kronecker, que nous adopterons dans cette section :

A system is solved if each root is represented in such way as to allow the performance of any arithmetical operations over the arithmetical expressions of its coordinates

Cette caractérisation est toutefois incomplète puisqu'une telle paramétrisation ne tient pas compte des multiplicités des racines [90]. Une idée naturelle est alors d'établir une relation directe entre les multiplicités des racines de f et celles des zéros de $\mathcal{I}$:

Définition 3.11. ***FR**-[114] Soient $\mathcal{I}$ un idéal de $\mathbb{Q}[X_1, ..., X_n]$, f un polynôme de $\mathbb{Q}[T]$ et ϕ: $\boldsymbol{V}(\mathcal{I}) \longrightarrow \boldsymbol{V}(f)$ un isomorphisme d'ensembles algébriques. On dit que (ϕ, f) est une Représentation Univariée de $\mathcal{I}$ si $\mu(\alpha) = \mu(\phi(\alpha))$. On dira que (ϕ, f) est une Représentation Univariée Rationnelle (RUR) si ϕ^{-1} coïncide sur $\boldsymbol{V}(f_t)$ avec $\beta \mapsto (\psi_1(\beta), ..., \psi_n(\beta))$, avec $\psi_i \in \mathbb{Q}(T)$, $i = 1...n$.*

A la différence d'une paramétrisation rationnelle quelconque de $\boldsymbol{V}(\mathcal{I})$, la notion de Représentation Univariée Rationnelle de $\mathcal{I}$ cache donc quelques propriétés algébriques que l'on peut mettre un peu plus en évidence :

Proposition 3.12. ***FR**-[114] Pour que (ϕ, f) soit une Représentation Univariée de $\mathcal{I}$, il faut et il suffit que :*

- *il existe t séparant $\boldsymbol{V}(\mathcal{I})$ tel que f soit le polynôme caractéristique de m_t*
- *il existe un morphisme $\Phi^\phi\text{:}\, \mathbb{Q}[T] \longrightarrow \mathbb{Q}[X_1, ..., X_n]$ tel que*
  - *Φ^ϕ représente ϕ ($\forall \alpha \in \boldsymbol{V}(\mathcal{I})$, $\Phi^\phi(T)(\phi(\alpha)) = \alpha$)*
  - *$\forall P \in \mathbb{Q}[T]$, $\mathrm{Trace}(P) = \mathrm{Trace}(\Phi^\phi(P))$ (avec les notations du théorème 3.10).*

Le calcul d'une *Représentation Univariée Rationnelle* de $\mathcal{I}$ se décompose donc naturellement en deux sous problèmes : trouver un élément séparant $\boldsymbol{V}(\mathcal{I})$, puis, étant donné cet élément séparant, trouver un morphisme de Φ^ϕ « préservant les traces », ou, de façon équivalente, un isomorphisme d'ensembles algébriques ϕ « préservant les multiplicités ». On dissocie ces deux prérequis en définissant, pour tout élément $t \in \mathbb{Q}[X_1, ..., X_n]$ (ne séparant pas nécessairement $\boldsymbol{V}(\mathcal{I})$) un *candidat Représentation Univariée Rationnelle* :

Définition 3.13. ***FR-[117]*** *On appelle candidat Représentation Univariée Rationnelle (candidat-RUR) de $\mathcal{I}$ la donnée d'un polynôme $f_t \in \mathbb{Q}[T]$ et de n de fractions rationnelles de $\mathbb{Q}(T)$ $f_{t,X_1}, ..., f_{t,X_n}$ tels que si t sépare $\boldsymbol{V}(\mathcal{I})$, (ϕ_t, f_t) est une Représentation Univariée Rationnelle de $\mathcal{I}$:*

$$\begin{array}{rccc} \phi_t\colon & \boldsymbol{V}(\mathcal{I}) & \longrightarrow & \boldsymbol{V}(f_t) \\ & \alpha=(\alpha_1,...,\alpha_n) & \rightarrowtail & t(\alpha) \\ & (f_{t,X_1}(\beta),...,f_{t,X_n}(\beta)) & \leftarrowtail & \beta \end{array}$$

Sous certaines hypothèses, un candidat-RUR peut être donné par une base de Gröbner de $\langle p_1, ..., p_s, T-t\rangle$ pour l'ordre lexicographique en supposant T plus petite que toutes les autres variables [15]. En effet, si le polynôme minimal de m_T dans $\frac{\mathbb{Q}[T,X_1,...,X_n]}{\langle p_1,...,p_s,T-t\rangle}$ est de même degré que la dimension de $\frac{\mathbb{Q}[X_1,...,X_n]}{\mathcal{I}}$ (qui est égale à celle de $\frac{\mathbb{Q}[T,X_1,...,X_n]}{\langle p_1,...,p_s,T-t\rangle}$), cette base sera de la forme $\{f_t(T), X_1 - f_1(T), ..., X_n - f_n(T)\}$. Cependant, on ne peut qualifier cette stratégie de calcul de candidat-RUR dans le cas général puisque si $\langle p_1, ..., p_s, T-t\rangle$ n'est pas *shape position*, la base n'aura pas cette forme.

Plus généralement, il n'existe que très peu d'algorithmes de calcul de candidat-RUR, l'essentiel des solutions proposées calculant en fait de simples paramétrisations rationnelles de $\boldsymbol{V}(\mathcal{I})$, autrement dit des candidat-RUR de $\sqrt{\mathcal{I}}$ ([56],[21],[107],...). En fait, la plupart permettent un calcul de RUR de $\sqrt{\mathcal{I}}$ probabiliste, en choisissant l'élément séparant t au hasard, les méthodes déterministes étant rares (garantie que t est séparant), très rares sont les méthodes déterministes efficaces en pratique : les principales seront présentées dans la section qui suit. Parmi les méthodes qui se rapprochent le plus du calcul d'une RUR, on peut citer [107] qui utilise une déformation infinitésimale ou [4] qui fournit une paramétrisation rationnelle pour chaque multiplicité possible.

Les trois résultats qui suivent montrent que l'on peut définir de façon universelle un candidat-RUR et le calculer avec les prérequis que nous nous sommes fixés (Notation 3.4). Le premier donne une définition brutale d'un candidat-RUR en dehors de toute préoccupation calculatoire :

Définition 3.14. ***FR-[113],FR-[114]*** *Soit $\mathcal{I} \subset \mathbb{Q}[X_1, ..., X_n]$ un idéal zéro-dimensionnel et t un polynôme de $\mathbb{Q}[X_1, ..., X_n]$ quelconque. On définit :*

- $f_t(T) = \prod_{\alpha \boldsymbol{V}(\mathcal{I})} (T - t(\alpha))^{\mu(\alpha)}$ *et* $\bar{f_t}(T)$ *sa partie sans facteurs carrés.*
- $g_t(T) = \frac{\partial \bar{f_t}}{\partial T}$
- $\forall v \in \mathbb{Q}[X_1, ..., X_n]$ $g_{t,v}(T) = \sum_{\alpha \in \boldsymbol{V}(\mathcal{I})} \mu(\alpha) v(\alpha) \prod_{\beta \in \boldsymbol{V}(\mathcal{I}), \beta \neq \alpha} (T - t(\alpha))$

Le théorème suivant montre que l'ensemble de polynômes ci-dessus est bien un candidat-RUR

Théorème 3.15. ***FR-[114]*** *Soit $\mathcal{I} \subset \mathbb{Q}[X_1, ..., X_n]$ un idéal zéro-dimensionnel. En reprenant les notations de la définition 3.13, pour tout polynôme $t \in \mathbb{Q}[X_1, ..., X_n]$, on pose :*

$$\begin{array}{rccc} \phi_t\colon & \boldsymbol{V}(\mathcal{I}) & \longrightarrow & \boldsymbol{V}(f_t) \\ & \alpha=(\alpha_1,...,\alpha_n) & \rightarrowtail & t(\alpha) \\ & (\frac{g_{t,X_1}(\beta)}{g_t(\beta)},...,\frac{g_{t,X_n}(\beta)}{g_t(\beta)}) & \leftarrowtail & \beta \end{array}$$

Alors, (ϕ_t, f_t) est un candidat-RUR de $\mathcal{I}$ et, de plus, $\mu(\alpha) = \frac{\frac{\partial \bar{f_t}(t(\alpha))}{\partial T}}{g_t(t(\alpha))}$ pour tout $\alpha \in \boldsymbol{V}(\mathcal{I})$.

Enfin, on obtient une expression équivalente mais cette fois-ci calculable du candidat-RUR :

Corollaire 3.16. ***FR-[114]*** *Soient $\mathcal{I} \subset \mathbb{Q}[X_1, ..., X_n]$ un idéal zéro-dimensionnel et $t \in \mathbb{Q}[X_1, ..., X_n]$ une forme linéaire. On définit :*

- $h_t(T) = \text{polchar}(m_t)$ *le polynôme caractéristique de m_t dans $\frac{\mathbb{Q}[X_1,...,X_n]}{\mathcal{I}}$ et $\bar{h_t}(T) = \sum_{i=0}^{d'} a_i T^{d'-i}$ sa partie sans facteurs carrés.*
- $H_j(T) = \sum_{i=0}^{j} a_i T^{j-i}$ *le j-ième polynôme de Horner associé à $\bar{h_t}(T)$;*

- $\forall v \in \mathbb{Q}[X_1, ..., X_n]$ $h_{t,v}(T) = \sum_{i=0}^{d'-1} \text{Trace}(v\, t^i)\, H_{d'-i-1}(T)$

Alors, $\left(h_t, \frac{h_{t,X_1}}{h_{t,1}}, ..., \frac{h_{t,X_n}}{h_{t,1}}\right)$ *est un candidat RUR. Plus précisément, en reprenant les notations de la définition 3.13, si* t *sépare* $\boldsymbol{V}(\mathcal{I})$, $d' = \#\boldsymbol{V}(\mathcal{I})$, $h_t = f_t$, $h_{t,1} = g_t$ *et* $h_{t,v} = g_{t,v}, \forall v \in \mathbb{Q}[X_1, ..., X_n]$.

A la différence d'une paramétrisation rationnelle standard (par exemple [107],[56]), la représentation univariée rationnelle d'un idéal permet de ne pas perdre l'information sur les multiplicités des solutions. En fait, on peut même calculer une fraction rationnelle supplémentaire donnant explicitement la multiplicité des racines de l'idéal étudié :

Corollaire 3.17. ***FR****-[113], **FR**-[114] pour tout* $\alpha \in \boldsymbol{V}(\mathcal{I})$, $\mu(\alpha) = \frac{\frac{\partial \text{squarefreepart}(f_t(T))}{\partial T}}{\text{squarefreepart}(\frac{\partial (f_t(T))}{\partial T})}$ *ou* squarefreepart(.) *désigne la partie sans facteurs carrés d'un polynôme.*

Comme indiqué plus haut, calculer un candidat-RUR ne résout que partiellement le problème, encore faut-il trouver un élément séparant pour obtenir au final une RUR. On peut imaginer deux situations : soit calculer séparément une forme linéaire séparant $\boldsymbol{V}(\mathcal{I})$ comme par exemple en combinant le lemme 3.9 et le théorème 3.10, soit choisir une forme linéaire t arbitrairement et vérifier qu'elle est séparante, connaissant un candidat-RUR :

Lemme 3.18. ***FR****-[113], **FR**-[114] Un candidat-RUR* $\{f_t, g_{t,1}, g_{t,X_1}, ..., g_{t,X_n}\}$ *est une RUR si et seulement si les polynômes* $f_t(t), g_{t,1}(t)X_i - g_{t,X_i}(t), i = 1...n$ *appartiennent à* $\sqrt{\mathcal{I}}$.

3.4 Algorithmes et complexité

Au regard du corollaire 3.16, le calcul d'un candidat-RUR se résume à celui du polynôme caractéristique de m_t et aux traces $\text{Trace}(m_{X_j t^i})$, $j = 1...n, i = 1...d$, avec $d = \#\boldsymbol{V}(\mathcal{I})$. En remarquant que les traces $\text{Trace}(m_{t^i})$, $i = 1...D$, avec $D = \dim(\frac{\mathbb{Q}[X_1, ..., X_n]}{\mathcal{I}})$ sont en fait les sommes de Newton de f_t (d'après le théorème 3.3) on peut obtenir un candidat-RUR en $O(n\, D^2)$ opérations arithmétiques connaissant les traces $\text{Trace}(X_j t^i)$, $j = 1...n$, $i = 1...D$ et $\text{Trace}(t^i)$, $i = 1...D$ **FR**-[114] :

Algorithm RUR-CANDIDATE

- **Input** : t, $\text{Trace}(t^i), i = 1...D$, $\text{Trace}(X_j t^i), i = 1...D-1, j = 1...n$
- **Output** : $f_t, g_t, g_{t,X_1}, ...g_{t,X_n}$
- Solve the triangular linear system $\left\{(D - i)\ a_i = \sum_{j=0}^{i-1} a_{i-j}\text{Trace}(t^j)\right\}_{i=0...D}$ and set $f_t(T) = \sum_{i=0}^{D} a_i T^{D-i}$
- Compute $\bar{f_t}(T) = \sum_{i=0}^{d} b_i T^{d-i}$,the squarefree part of f_t and set $H_j(T) = \sum_{i=0}^{j} b_i T^{j-i}$
- Set $g_{t,1}(T) = \frac{\partial \bar{f_t}}{\partial T}$
- for $j = 1...n$ compute $g_{t,X_j}(T) = \sum_{i=0}^{d} \text{Trace}(X_j\, t^i) H_{D-j-1}(T)$

On peut remarquer que si $d < D$ (cas de systèmes avec racines multiples), il est possible de calculer moins de traces ($\text{Trace}(X_j t^i), i = 1...d-1$ au lieu de $i = 1...D$) mais nous garderons cette version (qui est de même complexité dans le pire cas) pour simplifier les descriptions à venir.

3.4.1 Calcul d'un candidat RUR - algorithme CLASSIC

On ne peut raisonnablement pas espérer calculer un polynôme d'un candidat-RUR en moins de $O(D^2)$ opérations arithmétiques dans le cas général, ne serait-ce que parce que l'un d'eux est le polynôme caractéristique d'une matrice a priori sans structure particulière. L'étape principale dans le calcul d'un candidat-RUR est donc le calcul des traces $\mathrm{Trace}(t^i), i=1...D$, $\mathrm{Trace}(X_j t^i)$, $i=1...D, j=1...n$.

Notation 3.19. *En utilisant les notations 3.4, on désigne par* $\bar{h}[i]$ *la i-ième coordonnée de* $\bar{h} \in \frac{\mathbb{Q}[X_1,...,X_n]}{\mathcal{I}}$ *dans* $\mathcal{B}=\{w_1,...,w_D\}$ *et, pour tout polynôme* $P \in \mathbb{Q}[X_1,...,X_n]$*, par* M_P *la matrice de* m_P *dans la base* $\mathcal{B}$*. Pour toute matrice M,* $M[1]$ *désignera sa première ligne.*

Calculer $\mathrm{Trace}(m_p)$ revient à calculer $\sum_{i=0}^{D} \overline{Pw_i}[i]$ ou encore $\sum_{j=0}^{D}\sum_{i=0}^{D} a_j \overline{w_j w_i}[i]$ si $\overline{P} = \sum_{i=0}^{D} a_i w_i$, ce qui se réécrit de façon plus compacte $\mathrm{Trace}(m_P) = Q_1\ \bar{P}$ ou $Q_1 = [\mathrm{Trace}(w_i w_j)]_{i=1...D}^{j=1...D}$ est la matrice de la forme quadratique q_1 (Théorème 3.10). On peut immédiatement constater que l'on évitera des calculs aussi coûteux que redondants si l'on pré-calcule tous les produits $\overline{w_i w_j}$.

Définition 3.20. *Avec les notations 3.4, on note* $\mathcal{T}=\{\overline{w_i w_j}, i=1...D, j=1...D\}$*, la table de multiplication de* $\frac{\mathbb{Q}[X_1,...,X_n]}{\mathcal{I}}$ *(pour la base* $\mathcal{B}$*)*

La table $\mathcal{T}$ peut se calculer en $O(\#\mathcal{T}\, D^2)$ opérations arithmétiques, en utilisant le fait que $w_1=1$ et que $\forall j=1...D, \exists k, w_j = X_k w_i$ (Notation 3.4) **FR**-[117]. Nous garderons par la suite $\#\mathcal{T}$ comme paramètre de complexité car cette valeur oscille entre $O(D)$ (par exemple dans le cas univarié ou *shape position*) et $O(D^2)$ (borne supérieure naïve et jamais atteinte). Nous reviendrons sur le coût binaire de ce calcul ainsi que l'occupation mémoire engendrée plus loin: nous supposons pour l'instant que $\mathcal{T}$ est connue.

Comme première utilisation de $\mathcal{T}$, l'algorithme qui suit **FR**-[113], **FR**-[117], calcule la matrice Q_1 de la forme quadratique q_1 (voir Théorème 3.10) qui nous servira à rendre plus compactes les notations dans les algorithmes qui suivront :

Algorithm Hermite

- **Input** : $\mathcal{T}$
- **Output** : $Q_1 = [\mathrm{Trace}(w_i w_j)]_{i=1...D}^{j=1...D}$
- for $i=1...D$ do $\mathrm{Trace}(w_i) = \sum_{j=0}^{D} \overline{w_i w_j}[i]$
- for $\overline{w_i w_j} \in \mathcal{T}$ do $\mathrm{Trace}(w_i w_j) = \overline{w_i w_j} \cdot Q_1[1]$

On peut voir qu'en termes d'opérations arithmétiques, la construction de Q_1 n'est pas coûteuse comparée à celle de la table de multiplication. L'algorithme qui suit permet le calcul des traces requises pour celui d'un candidat-RUR et ne diffère de la solution proposée dans **FR**-[114] que par l'utilisation de $Q_1 = [\mathrm{Trace}(w_i w_j)]_{i=1...D}^{j=1...D}$ qui ne sert que de raccourci :

Algorithm Traces-Classic

- **Input** : $\mathcal{T}$, $Q_1 = [\mathrm{Trace}(w_i w_j)]_{i=1...D}^{j=1...D}$, (and thus $M_t, M_{X_1}, ..., M_{X_n}$)
- **Output** : $\mathrm{Trace}(t^i w_j), i=0...D, j=1...D$, $\mathrm{Trace}(X_j t^i), i=1...D, j=1...n$
- set $t^0 = [1,0,...,0]$
- for $i=1...D$ do
  - $t^{\bar{i}} = M_t\, \overline{t^{i-1}}$
  - $[\mathrm{Trace}(t^i w_1), ..., \mathrm{Trace}(t^i w_D)] = Q_1\, t^{\bar{i}}$
  - for $j=1...D$ $\mathrm{Trace}(X_j t^i) = \overline{X_j}\,[\mathrm{Trace}(t^i w_1), ..., \mathrm{Trace}(t^i w_D)]$

On peut voir ainsi qu'il en coûte $O(D^3 + n\ D^2)$ opérations arithmétiques pour calculer les traces nécessaires à l'algorithme RUR-CANDIDATE par cette fonction et $O(\#\mathcal{T} D^2 + D^3 + n\ D^2)$ opérations arithmétiques si l'on inclut le calcul de $\mathcal{T}$ et de Q_1.

3.4.2 Complexité binaire de l'algorithme CLASSIC

Comme évoqué en introduction, la complexité arithmétique ne donne qu'une idée toute relative de l'efficacité pratique d'un algorithme. Ce chapitre en est par ailleurs l'illustration puisqu'un calcul de complexité binaire inversera les tendances annoncées par les calculs de complexité arithmétique.

Il s'agit essentiellement de prendre en compte les phénomènes de croissance de coefficients dans le coût des opérations de base. Nous reprenons ici le modèle utilisé dans **FR**-[113] et **FR**-[114] puis affiné dans **FR**-[117]. En résumé, les opérations les plus coûteuses dans le calcul de candidat-RUR sont des itérations de type $v_i = M\ v_{i-1}$, v_0 étant un vecteur et M une matrice carrée. Une telle opération sera notée Mv-chain(i), par la suite, i étant le nombre d'itérations.

Dans toute cette section, on utilise les conventions suivantes :

Notation 3.21. *On suppose que la taille des coefficients apparaissant dans les matrices de multiplication m_{X_i} est $O(s)$, celles-ci étant de dimension D. On note δ le degré total maximal des $w_i \in \mathcal{B}$.*

En modifiant les structures de données, précisément en représentant les matrices (resp. vecteurs) à coefficients rationnels par un dénominateur commun aux entrées et une matrice (resp. vecteur) à coefficients entiers, le modèle de complexité utilisé donne le tableau suivant pour le calcul Mv-chain(i): s_M, s_0 et s_i sont les tailles des entiers intervenant respectivement dans M, v_0 et v_i, la 3-ième colonne donne le nombre d'opérations binaires réalisées et la 4-ème le nombre d'opérations arithmétiques.

Comp.	Entrée	Sortie	op. bin.	op. arith.
$\mathrm{Mv-Chain}\,(i)$	$s_M,\ s_0$	$s_i = O(j(s_M + D) + s_0)$	$O(D^2 j(s_0 + j(s_M + D)))$	$O(i\,D^2)$

Deux remarques concernant ces résultats :

- les coûts binaires sont obtenus en utilisant le fait que la multiplication d'un entier de taille t par un entier de taille $t' \gg t$ est linéaire en t', ce qui cadre avec la réalité. Ils auraient pu être obtenus simplement, aux facteurs log près, en utilisant une multiplication rapide de type fft (ce qui est moins le reflet de la réalité).
- la taille des objets en sortie est atteinte à une petite constante près : il suffit de supposer que M et v_0 n'ont que des coefficients entiers positifs, ce qui correspond par exemple aux cas ou l'on calcule dans l'algèbre quotient à partir d'une base de Gröbner dont les polynômes sont unitaires et dont tous les coefficients, hormis les termes de tête, sont négatifs.

Le calcul de complexité binaire justifie un peu plus le choix qui a été fait de préserver $\#\mathcal{T}$ comme paramètre de complexité. En effet, c'est un exemple typique ou il faut dissocier la méthode de calcul du calcul de borne sur les coefficients apparaissant dans le résultat. Le calcul de cette table est itératif, puisque chaque élément de la table est obtenu en utilisant la propriété de connexité de $\mathcal{B}$: $w_1 = 1$ et , $\forall j = 1...D, \exists k\, w_j = X_k w_i$. Une entrée est donc le résultat d'une itération Mv-chain(i) ou i est majoré par 2δ, δ étant le degré maximal des éléments de $\mathcal{B}$. Les autres sous-algorithmes du calcul d'un candidat-RUR s'étudient facilement, et on obtient alors **FR**-[117] :

Comp.	output (coeffs)	binary op.	arith. op
$\mathcal{T}$	$O(\delta(s+D))$	$O(\sharp\mathcal{T} D^2 \delta(s+D))$	$O(\sharp\mathcal{T} D^2)$
HERMITE	$O(\delta(s+D))$	$O(\sharp\mathcal{T} D \delta(s+D))$	$O(\sharp\mathcal{T} D)$
TRACES-CLASSIC	$O(D(s+D))$	$O((D^4 + n\,D^3)(s+D))$	$O(D^3 + n\,D^2)$
RUR-CANDIDATE	$O(D\,(s+D))$	$O(n\,D^3(s+D))$	$O(n\,D^2)$

Une remarque sur la complexité de l'algorithme RUR-Candidate : la taille des coefficients de l'entrée est la même que la taille des coefficients de la sortie $O(D\ (s+D))$. Si l'on observe l'algorithme, la seule instruction pouvant entraîner une croissance mesurable est la résolution du système triangulaire : $\left\{(D-i)\,a_i = \sum_{j=0}^{i-1} a_{i-j}\mathrm{Trace}(t^j)\right\}_{i=0\ldots D}$. Mais on sait que les a_i sont les coefficients du polynôme caractéristique d'une matrice dont les coefficients sont de taille s, ceux-ci n'excèdent donc pas $O(D\,s)$.

Une autre remarque importante est que le calcul de $\mathcal{T}$ est toujours dominant si l'on ne compte que les opérations arithmétiques, mais ça n'est plus forcément vrai dès lors que l'on prend en compte la croissance des coefficients. Par exemple, lorsque l'idéal étudié est *shape position* et que l'on utilise une base de Gröbner lexicographique pour le représenter, $\delta = D = O(\#\mathcal{T})$. Dans le cas ou l'idéal est défini par n polynômes en une variable de degré δ, $\#\mathcal{T} = D = O(\delta^n)$ et le calcul de $\mathcal{T}$ n'est toujours pas dominant. En travaillant un peu, on peut même étendre ce dernier résultat aux situations généralement considérées en complexité, par exemple, le cas d'idéaux intersection complète, en jouant sur le choix d'une base et sur les majorations δ et D, mais tout ceci n'a pas vraiment de contre-partie pratique. On retiendra simplement que le calcul de $\mathcal{T}$ ne pollue pas forcément la complexité globale du calcul d'un candidat-RUR.

3.4.3 Calcul d'un candidat RUR - l'algorithme BS/GS et sa complexité

Dans l'algorithme Traces-Classic, on peut remarquer que l'on pourrait faire l'économie du calcul des vecteurs (ou pour le moins d'une partie de ces vecteurs) $[\mathrm{Trace}(t^i w_1), ..., \mathrm{Trace}(t^i w_D)]$ en pré-calculant les vecteurs $[\mathrm{Trace}(X_i w_1),\ ...,\ \mathrm{Trace}(X_i w_D)]$ et en remplaçant la principale boucle par :

- for $i = 1...D$ do
  - $\overline{t^i} = M_t\,\overline{t^{i-1}}$
  - $\mathrm{Trace}(t^i) = Q_1[1]\,\overline{t^i}$
  - $j = 1...D$ $\mathrm{Trace}(X_j\,t^i) = \overline{t^i}\ [\mathrm{Trace}(X_i w_1), ..., \mathrm{Trace}(X_i w_D)]$

On aurait ainsi l'impression de mieux exploiter les quantités calculées, les 2-ième et 3-ième lignes générant $O(n\ D^2)$ opérations arithmétiques contre $O(D^3)$ précédemment. Ceci dit, l'opération dominante de la boucle reste la 1ère ligne dans les deux cas et on a alourdit le calcul des prérequis ainsi que l'occupation mémoire de l'ensemble (pré-calcul et stockage de $[\mathrm{Trace}(X_i w_1)$, ..., $\mathrm{Trace}(X_i w_D)]$).

Dans [19], les auteurs proposent un algorithme probabiliste permettant de calculer une paramétrisation rationnelle de n'importe quel système zéro-dimensionnel utilisant une stratégie de type « pas de bébé/pas de géant » permettant de réduire le poids du calcul des puissances de t ($\overline{t^i} = M_t\,\overline{t^{i-1}}$, $i = 1...D$). L'algorithme proposé est « doublement probabiliste » mais fournit un objet proche d'un candidat-RUR. Il dépend d'un choix arbitraire de forme linéaire séparante mais ne propose aucun moyen de valider ce choix, que ce soit a priori ou a posteriori, ce qui rapproche l'objet calculé d'un candidat-RUR, au détail près que l'information sur les multiplicités n'est pas forcément préservée. Il dépend également d'un choix arbitraire de forme linéaire permettant d'appliquer l'algorithme de Berlekamp-Massey pour le calcul de f_t (ou pour le moins d'un diviseur de f_t). En fait, les auteurs utilisent une généralisation des formules de la proposition 3.16 substituant une forme linéaire « suffisamment générique » à celle de l'application Trace. Au regard de la proposition 3.12, utiliser une forme linéaire différente semble être équivalent à utiliser l'application Trace avec un autre élément séparant si l'on désire calculer un candidat-RUR. Comme montré dans **FR**-[117], si l'on force l'utilisation de l'application Trace pour la formulation du résultat, l'utilisation de l'algorithme de Berlekamp-Massey ne se justifie plus. Au final on ne retiendra de [19] que l'idée d'utiliser la stratégie « pas de bébé - pas de géant », essentiellement dérivée de [131] et [20]. Notons que la borne de complexité annoncée dans [19] ($O(n2^n D^{5/2})$ opérations arithmétiques) n'est correcte que dans certains cas d'idéaux intersection complète, mais on peut toutefois facilement vérifier que l'algorithme réalise en fait $O(n\#\mathcal{T}D^{3/2})$ opérations arithmétiques si l'on suppose connue $\#\mathcal{T}$.

Dans ce qui suit, on propose deux variantes **FR**-[117] qui ne diffèrent que par la façon dont est calculée la multiplication d'un élément $\bar{p} \in \frac{\mathbb{Q}[X_1,...,X_n]}{\mathcal{I}}$ par $t^{\bar{k}}$, avec $k = \left\lfloor \frac{D}{\sqrt{D}} \right\rfloor$.

La première utilisera une fonction notée MULTK-A qui est une version épurée de la méthode utilisée dans [19] : on écrit le produit comme une combinaison linéaire d'éléments de $\mathcal{T}$, $\overrightarrow{t^k p} = \sum_{m \in \mathcal{T}} a_m \vec{m}$, puis on calcule cette combinaison linéaire de $\#\mathcal{T}$ vecteurs de dimension D.

La deuxième utilisera une fonction notée MULTK-B qui suppose connue la matrice M_{t^k} de l'application m_{t^k} (qui devient donc un nouveau prérequis à calculer séparément) et se contente alors de calculer le produit matrice/vecteur $M_{t^k}\bar{p}$.

L'algorithme général qui suit **FR**-[117] prend en paramètre une de ces fonctions et combine les stratégies de **FR**-[114] et [19] :

Algorithm TRACES-BS/GS

- Input : $\mathcal{T}$, Q_1, MULTK: $\bar{p} \longrightarrow \overline{t^k p}$
- Set $k = \frac{D}{\left\lfloor \sqrt{D} \right\rfloor}, k' = \frac{D}{k}$;
- [B-1] set $t^0 = [1, 0, ..., 0]$ for $i = 1...k$ $\overline{t^i} = M_t \overline{t^{i-1}}$
- [B-2] for $j = 1...n$, $i = 1...k$ do $\overline{X_j t^i} = M_{X_j} \overline{t^i}$
- for $i = 1...k'$
  - [B-3] $\overline{t^{ki}} = \text{MULTK}(\overline{t^{k(i-1)}})$
  - [B-4] $[\text{Trace}(t^{ki} w_1), ..., \text{Trace}(t^{ki} w_D)] = Q\,\overline{t^{ki}}$
  - [B-5] for $j = 1...k$ $\text{Trace}(t^{ki+j}) = \overline{t^j}\,[\text{Trace}(t^{ki} w_1), ..., \text{Trace}(t^{ki} w_D)]$
    - [B-6] for $u = 1...n$ $\text{Trace}(X_u t^{ki+j}) = \overline{X_u t^j}\,[\text{Trace}(t^{ki} w_1), ..., \text{Trace}(t^{ki} w_D)]$

En utilisant exactement le même modèle et le mêmes astuces de calcul que pour l'algorithme CLASSIC, on obtient facilement les complexités des sous-algorithmes utilisés dans le calcul d'un candidat-RUR pour chacune des stratégies de type « pas de bébé/pas de géant » (les calculs sont détaillés dans **FR**-[117]) :

Comp.	output (coeffs)	binary op.	arith. Op.
$\mathcal{T}$	$O(\delta(s+D))$	$O(\sharp\mathcal{T} D^2 \delta(s+D))$	$O(\sharp\mathcal{T} D^2)$
HERMITE	$O(\delta(s+D))$	$O(\sharp\mathcal{T} D \delta(s+D))$	$O(\sharp\mathcal{T} D)$
M_{t^k}	$O((k+\delta)(s+D))$	$O(D^3\,(k+\delta)\,(s+D))$	$O(D^3)$
TRACES-BS/GS (MULTK-A)	$O(D(s+D))$	$O(D^{5/2} \sharp\mathcal{T}\,(s+D))$	$O(\sharp\mathcal{T} D^{3/2} + n\,D^2)$
TRACES-BS/GS (MULTK-B)	$O(D(s+D))$	$O((D^3\,(D^{1/2}+n+\delta)(s+D))$	$O(D^3 + n\,D^2)$
RUR-CANDIDATE	$O(D\,(s+D))$	$O(n\,D^3(s+D))$	$O(n\,D^2)$

3.4.4 Choix d'une stratégie pour le calcul d'un candidat-RUR

Dans ce qui suit, on note CLASSIC le calcul d'un candidat-RUR utilisant l'algorithme TRACES-CLASSIC et BS/GS-A (resp. BS/GS-B) l'algorithme calculant un candidat-RUR en utilisant l'algorithme TRACES-BS/GS et la stratégie MULTK-A (resp. MULTK-B).

On peut voir que la stratégie BS/GS-A, bien qu'étant déterministe et calculant un candidat-RUR, améliore la complexité arithmétique de l'algorithme [19] qui est probabiliste et perd l'information sur les multiplicités : $O(\#\mathcal{T} D^{3/2} + n\,D^2)$ ici contre $O(n\#\mathcal{T} D^{3/2})$ dans [19].

Mais, même si la stratégie BS/GS-A est celle qui réclame éventuellement le moins d'opérations arithmétiques (au moins pour certaines classes d'exemples), elle est toujours plus lente que la stratégie BS/GS-B si l'on prend en compte la croissance des coefficients. Une explication simple est que dans le cas de BS/GS-B, les sous-algorithmes coûteux en termes d'opérations arithmétiques (par exemple le calcul préalable de M_{t^k}) sont utilisés très tôt dans le processus global, les opérations arithmétiques induites sont donc effectuées sur des scalaires de plus « petite » taille.

Enfin, on peut noter que l'algorithme BS/GS-B est, en théorie, uniformément le plus rapide des trois algorithmes étudiés dans ce chapitre pour le calcul d'un candidat-RUR.

Il est difficile de mesurer le poids réel du calcul de $\mathcal{T}$ dans le processus global, puisque que cela revient à comparer $\#\mathcal{T}\delta$ qui oscille entre δD (classe d'exemples favorable) et δD^2 (borne naïve a priori jamais atteinte) avec $D^{3/2} + n\ D$ (BS/GS-B) ou D^2 (classic), mais on peut s'attendre à ce que ce soit l'opération la plus coûteuse, au moins dans le cas de BS/GS-B. Nous reviendrons plus tard sur l'aspect « occupation mémoire » élément clé pour jauger de l' « efficacité » de ces méthodes.

3.4.5 RUR et candidat-RUR

Pour l'instant, nous ne savons « que » calculer un candidat-RUR, l'objectif ultime (calcul d'une RUR) ne sera atteint que si l'on résout également le problème de l'élément séparant. Il y a deux possibilités : soit celui-ci est connu avant le calcul de RUR, soit on choisit une forme linéaire quelconque pour calculer le candidat-RUR, auquel cas il faudra vérifier a posteriori que celle-ci sépare $\boldsymbol{V}(\mathcal{I})$ pour lever toute ambiguïté sur le résultat.

3.4.5.1 Calculer un élément séparant sans connaître de candidat-RUR

Conséquence directe des lemmes 3.8, 3.9 et du théorème 3.10, l'algorithme suivant permet de calculer une forme linéaire séparant $\boldsymbol{V}(\mathcal{I})$:

Algorithm FindSep-Classic

Input : $Q_1, M_{X_i}, i = 1...n$
Output : a linear form t which separates $\boldsymbol{V}(\mathcal{I})$

- $d := \text{rank}(Q_1)$
- for $i = 1...n^{\frac{d\,(d\,-\,1)}{2}}$
  - $M_t := \sum_{j=0}^{n} i^j\, M_{X_i}$
  - compute $f_t = \text{charpol}(M_t)$
  - compute $\bar{f_t}$ the squarefree part of f_t
  - if degree$(\bar{f_t}) = d$ then return(t)

Une remarque est que charpol(M_t) peut être calculé de la même façon que dans l'algorithme RUR-Candidate, c'est à dire à partir des Trace(t^i), $i = 1...D$ ou encore être remplacé par le polynôme minimal de M_t. Cette ligne de calcul ne sera donc jamais plus coûteuse que le calcul d'un candidat-RUR.

Une seconde remarque est qu'on obtient, en pratique, un élément séparant après 1 ou 2 tentatives. On distinguera donc la complexité théorique qui nécessite $n^{\frac{d(d\,-\,1)}{2}}$ tentatives dans le pire cas, de la complexité pratique ou « en moyenne » qui est de l'ordre du coût des calculs présents dans la boucle.

L'opération coûteuse de cet algorithme est en fait le calcul du rang de Q_1 (rank(Q_1)). En termes de nombre d'opérations arithmétiques, il est en $O(D^3)$ ce qui est équivalent au nombre d'opérations arithmétiques effectuées pour un calcul de candidat-RUR. Mais si Q_1 est de dimension D et admet des coefficients de taille s', alors la taille des coefficients apparaissant dans sa forme réduite (calcul de son rang) est $O(D\ s')$ **FR**-[112]. Ce phénomène est observable en pratique (12 exemples provenant de **FR**-[113]):

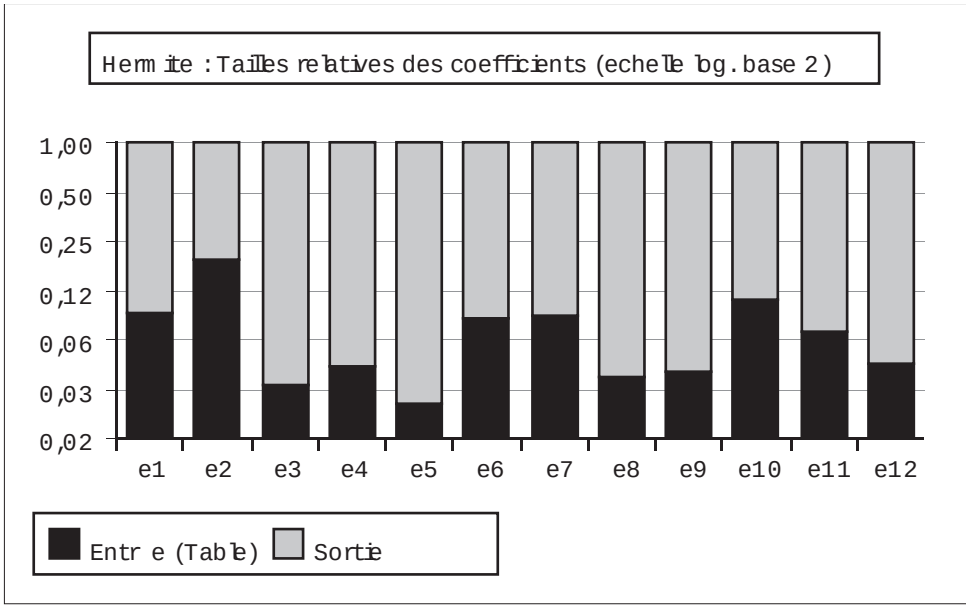


Figure 3.1. Taille des coefficients dans Q_1 avant et après réduction

Si l'on reporte le coût de cette réduction avec les tailles de coefficients que l'on a à traiter, on obtient alors **FR**-[114] :

Comp.	binary op.	arith. Op.
RANK(Q_1)	$O(D^4\delta(s+D))$	$O(D^3)$

Ainsi, si l'on garantit qu'un candidat-RUR est une RUR en calculant un élément séparant a priori par l'algorithme FINDSEP-CLASSIC, cette dernière opération est la plus coûteuse du processus global, même si l'on se contente d'observer la complexité « en moyenne ».

3.4.5.2 Vérifier si un élément est séparant connaissant un candidat-RUR

En combinant le lemme 3.18 et le théorème 3.10, on obtient un critère simple, réutilisant une partie des calculs déjà effectués, pour vérifier si un candidat-RUR est une RUR :

Corollaire 3.22. ***FR**-[113], **FR**-[114] Un candidat-RUR $\{f_t,\ g_{t,1},\ g_{t,X_1},\ ...,\ g_{t,X_n}\}$ est une RUR si et seulement si $Q_1\ \overline{f_t(t)} = [0, ..., 0]$, $Q_1\overline{g_{t,1}(t)X_i - g_{t,X_i}(t)} = [0, ...0]$, $i = 1...n$ avec $Q_1 = [\text{Trace}(w_i\, w_j)]_{i=1...D}^{j=1...D}$.*

L'algorithme qui suit **FR**-[117], réalise exactement les calculs suggérés par ce corollaire, la seule astuce additionnelle étant la manière dont est calculé $\overline{f_t(t)}$: on calcule en fait itérativement les polynômes de Horner ($\overline{H_i(t)}$, $i = 0...D$) qui lui sont associés pour faciliter celui des vecteurs $\overline{g_{t,1}(t)X_i - g_{t,X_i}(t)}$:

Algorithm CheckRUR-Classic

Input : $Q_1, f_t, M_t, M_{X_i}, \text{Trace}(t^j), \text{Trace}(X_i t^j), i = 1...n, j = 1...D$
Output : FALSE iff t is not a separating element.

- $\overline{H_0} = a_0[1, 0, ..., 0]$;
- for $i = 1...D$ $\overline{H_i(t)} = M_t \overline{H_{i-1}(t)} + a_i \overline{H_{i-1}(t)}$;
- if $Q_1 \overline{H_D(t)} \neq [0, ..., 0]$ then return(FALSE);
- $\overline{g_{t,1}(t)} = \sum_{i=1}^{D} \text{Trace}(t^j) \overline{H_{D-i-1}(t)}$
- for $i = 1...n$
  - $\overline{g_{t,X_i}(t)} = \sum_{i=1}^{D} \text{Trace}(X_i t^j) \overline{H_{D-i-1}(t)}$
  - if $Q_1(M_{X_i} \overline{g_{t,1}(t)} - \overline{g_{t,X_i}(t)}) \neq [0, ..., 0]$ then return(FALSE)
- return(TRUE)

Si l'on calcule la complexité de cet algorithme on obtient cette fois les mêmes complexités (arithmétique et binaire) que pour la fonction Traces-Classic (**FR**-[117]):

Comp.	binary op.	arith. Op.
CheckRUR-Classic	$O((D^4 + n\,D^3)(s + D))$	$O(D^3 + n\,D^2)$

En d'autres termes, vérifier qu'un candidat-RUR est une RUR est définitivement une opération difficile qui cependant n'est, en théorie, pas plus coûteuse que le calcul d'un candidat-RUR.

Dans le cas de l'utilisation de l'algorithme Classic pour le calcul d'un candidat-RUR, on peut se contenter de la fonction CheckRUR-Classic, même en pratique. Par contre, cette opération est la plus coûteuse en théorie si l'on utilise la stratégie BS/GS-B. Dans ce qui suit, on réutilise le principe de « produit transposé » [20] qui est à la source de [131] ou encore de [19] pour calculer $\overline{p(t)}$ pour un polynôme $\forall p, t \in \mathbb{Q}[X_1, ..., X_n]$. Notons que l'algorithme suivant **FR**-[117] prend comme paramètre une fonction MulTk qui peut être MulTk-A, MulTk-B ou n'importe quelle fonction permettant de multiplier par t^k modulo $\mathcal{I}$.

Algorithm ReducedEval

- **Input** : $k, 1, \bar{t}, ..., \overline{t^k}$, mulTk(u) and $p = \sum_{i=0}^{D} p_i T^i$
- **Output** : $\overline{p(t)} \in \mathbb{Q}[X_1, ..., X_n]/I$ (as a vector w.r.t. $\mathcal{B}$)
- Set $u = [0, ..., 0]$, $k' = \frac{D}{k}$
- for $i = k'...0$ do $u =$ mulTk $(u) + \sum_{j=0}^{k} p_{ki+j} \overline{t^j}$
- return(u)

La fonction ReducedEval effectue clairement $O(D^{1/2} T_k)$ opérations, T_k étant le nombre d'opérations que nécessite mulTk. Munis de ce « produit transposé », on peut faire évoluer la fonction de vérification d'un candidat-RUR **FR**-[117] :

Algorithm CHECKRUR-BS/GS

- **Input :** $f_t, g_t, g_{t,1}, ..., g_{t,n}$ a RUR-Candidate, $\mathcal{B}, t, Q$
- **Output :** TRUE if and only if the RUR-Candidate is a RUR
- $\overline{f_t(t)} =$ REDUCEDEVAL(f_t),
- if $Q.\overline{f_t(t)} \neq [0, ..., 0]$ then return(FALSE);
- $\overline{g_{t,1}(t)} =$ REDUCEDEVAL(g_t);
- for $i = 1...n$ do
  - $\overline{g_{t,i}(t)} =$ REDUCEDEVAL$(g_{t,i})$;
  - if $Q_1(\, M_{X_i}\overline{g_{t,1}(t)} - \overline{g_{t,X_i}(t)}) \neq [0, ..., 0]$ then return(FALSE)
- return(TRUE)

En appliquant toujours le même modèle de complexité, on obtient (**FR**-[117]) :

Comp.	binary op.	arith. Op.
CHECKRUR-BS/GS (MULTK-B)	$O(n\, D^{7/2}(s+D))$	$O(n\, D^{5/2})$

Dans le cas de l'utilisation des fonctions CHECKRUR-BS/GS et RURCANDIDATE-BS/GS avec la stratégie MULTK-B, nous n'avons donc pas réussi à rendre la vérification moins coûteuse que le calcul du candidat-RUR, celle-ci devient même l'opération dominante, ce qui relativise l'apport réel théorique des techniques de type « pas de bébé/pas de géant » et confirme encore une fois que certifier un élément séparant l'opération principale d'un algorithme non probabiliste.

Au final, on peut proposer un algorithme complet déterministe pour le calcul de RUR dont la complexité théorique est m fois la complexité la complexité théorique du calcul d'un candidat-RUR et de la vérification de l'élément (supposé) séparant qui lui est associé, m étant le nombre d'essais nécessaires pour « tirer » un élément séparant. En pratique, 1 essai suffira, presque toutes les formes linéaires étant séparantes.

On termine cette section en remarquant que la conjonction de RUR-Candidate-BS/GS et de CHECKRUR-BS/GS tous deux appliqués avec MULTK-B donne un algorithme déterministe pour le calcul d'un candidat-RUR accompagné d'un test de validation pour l'élément séparant (l'algorithme calcule donc une RUR ou renvoie une erreur si la forme linéaire n'est pas séparante) avec une complexité équivalente à l'algorithme proposé dans [19] qui lui est probabiliste, ne vérifie pas l'élément séparant et perd éventuellement l'information sur les multiplicités.

3.5 Un algorithme efficace pour le calcul de RUR

Les bornes supérieures fournies pour la complexité binaire des différentes stratégies permettant le calcul d'un candidat-RUR et la validation du choix d'une forme linéaire faisant office d'élément séparant incitent à favoriser une stratégie de type « pas de bébé/pas de géant ». La différence théorique avec l'algorithme CLASSIC est suffisamment faible pour que l'on ne fasse pas de choix aussi tranché pour l'implantation d'une méthode. En particulier, il est délicat de supposer, en pratique que $n \ll D^{1/2}$ et d'ignorer les constantes cachées dans la notation « $O()$ ».

A titre d'exemple, on peut se reporter au tableau du chapitre 2 mesurant le coût relatif de l'isolation des zéros de f_t par la méthode de Descartes : en prenant D pour le degré de ce polynôme et $D(D+s)$ comme taille de coefficients, c'est à dire les bornes sur le degré et la taille des coefficients de f_t obtenues ci-dessus, la borne supérieure de complexité pour ce calcul est alors $\tilde{O}(D^7(s+D)^2)$, ce qui est presque le carré de la meilleure borne obtenue pour la RUR. Pourtant, le chronomètre est sans appel, la part de l'isolation dans le processus global étant négligeable, en tout cas pour tous les exemples traités.

Plus sérieusement, on a pu constater, en pratique, quelques progrès liés à l'utilisation de l'algorithme BS/GS-B (comparativement à l'utilisation de l'algorithme Classic) sur des exemples admettant peu de variables et beaucoup de racines pour le calcul d'un candidat-RUR. Mais, sur ces exemples, c'est généralement le calcul de $\#\mathcal{T}$ qui s'est avéré dominant. Aucune expérience ne montre donc que l'on change de classe d'exemples en injectant du « pas de bébé/pas de géant ».

Pour ce qui est du calcul de RUR (incluant la vérification de l'élément séparant), la stratégie Classic est uniformément plus efficace que ses concurrentes sur les exemples traités. En particulier, on a pu observer, sans toutefois pouvoir l'expliquer, que les coefficients apparaissant dans les polynômes $\overline{H_i(t)}$ sont petits, en pratique, comparativement aux autres scalaires apparaissant lors des calculs, rendant la vérification CheckRUR-Classic fort peu coûteuse comparativement au calcul du candidat-RUR.

3.5.1 Utilisation de calcul modulaire

Un phénomène qui n'apparaît pas dans le calcul de bornes de complexité est qu'une RUR obtenue en utilisant une des variables comme élément séparant admet des coefficients, en pratique, (beaucoup) plus petits qu'une RUR obtenue pour une forme linéaire choisie au hasard. On peut partiellement expliquer ceci par le fait que les matrices M_{X_i} sont un peu creuses, alors qu'une combinaison linéaire de ces matrices sera plus dense. On aimerait donc tenter en premier le calcul d'un candidat-RUR en prenant pour t une des variables, mais, autant il est vrai que presque toute forme linéaire est séparante, même en pratique, autant se serait une erreur de supposer, même en pratique, que les variables le sont.

Cette dernière observation incite à travailler un peu plus le choix d'un élément séparant pour pouvoir essayer des formes linéaires n'ayant pas forcément beaucoup de chance d'être séparantes. Dans **FR**-[113] et **FR**-[114], l'idée est d'utiliser du calcul modulaire. Le lemme suivant se démontre facilement en utilisant des bases de Gröbner et en appliquant les résultats de spécialisation présents dans [51],[53] ou encore [52], il a simplement été réécrit pour tenir compte des prérequis utilisés dans ce chapitre.

Lemme 3.23. ***FR**-[113], **FR**-[114] Soit $\mathcal{I} \subset \mathbb{Q}[X_1, ..., X_n]$ un idéal zéro-dimensionnel tel que $\mathcal{I} = \langle p_1, ..., p_s \rangle, p_i \in \mathbb{Z}[X_1, ..., X_n]$ et p un entier premier. Si p ne divise aucun des dénominateurs apparaissant dans les matrices M_{X_i} des m_{X_i} exprimées dans la base $\mathcal{B}$ (Notation 3.4), et si* $\operatorname{rank}(Q_1 \bmod p) = \operatorname{rank}(Q_1)$*, alors toute forme linéaire t telle que $t \bmod p$ sépare $\boldsymbol{V}(\mathcal{I}_p)$ sépare également $\boldsymbol{V}(\mathcal{I})$.*

L'idée est donc de chercher un élément séparant de $\mathcal{I}_p$ pour prédire un élément séparant de $\mathcal{I}$ par un calcul rapide (pas de croissance de coefficients dans $\frac{\mathbb{Z}}{p\mathbb{Z}}$). Le but est d'éviter d'avoir à réduire Q_1, ce calcul étant plus coûteux que celui d'un candidat-RUR ainsi que celui de la certification de l'élément séparant. L'algorithme qui suit calcule un élément probablement séparant, en cherchant, en priorité, parmi les variables. Il utilise trois principes en plus du lemme ci-dessus :

- si p est choisi arbitrairement et suffisamment grand (en pratique voisin de 2^{32}), alors il vérifie presque toujours les hypothèses du lemme précédent;
- presque toutes les formes linéaires sont séparantes;
- si le degré de la partie sans facteurs carrés de $\operatorname{polmin}(m_t)$ est le nombre de zéros de l'idéal étudié, alors t est séparant.

Il permet de filtrer efficacement et rapidement les idéaux shape position (pour lesquels le calcul de RUR est, en principe, beaucoup plus simple) en utilisant le fait que, pour t donné, si $\operatorname{degree}(\operatorname{polmin}(M_t \bmod p)) = D$, alors nécessairement $\operatorname{degree}(\operatorname{polmin}(M_t)) = D$.

Algorithm PREDICTSEPELT

- **Input :** $M_{X_i}, \mathcal{B}$
- **Output :** t a separating linear form, isShapePosition a boolean which is TRUE iff $\mathcal{I}$ is *shape position*
- (*) Choose p not dividing any denominator in M_{X_i}
- Choose t randomly
  - $f = \text{minpol}(M_t \bmod p)$
  - $d' = \text{degre}(\text{squarefreepart}(f))$
- for $t' \in \{X_1, ..., X_n\}$
  - $f = \text{minpol}(M_{t'})$
  - $d' = \text{degre}(\text{squarefreepart}(f))$
  - if $d' = d$ return(t',$d = D$);
- return(t,d')

La complexité de l'algorithme PREDICTSEPELT est $O(n\ D^3)$ opérations arithmétiques dans $\frac{\mathbb{Z}}{p\,\mathbb{Z}}$ et donc de $O(n\,D^3)$ opérations binaires, ce qui est très en dessous de la complexité du calcul d'un candidat-RUR ou encore de la vérification d'un candidat-RUR. Comme en pratique cet algorithme fournira quasi systématiquement un élément séparant, la complexité « en moyenne » du calcul d'un RUR sera alors celle du calcul d'un candidat-RUR plus celle de la vérification de l'élément séparant. Par rapport à un choix au hasard, l'amélioration apportée par cette stratégie est la possibilité de choisir l'élément séparant parmi les variables et donc d'avoir des expressions finales avec des coefficients de plus petite taille.

3.5.2 Le cas des idéaux *shape position*

Avec l'utilisation de l'algorithme PREDICTSEPELT, on peut détecter, disons filtrer, les idéaux *shape position*. Dans ce cas, calculer une RUR se ramène à la résolution d'un système linéaire de la forme $M\,X = B$ ou M est la matrice dont les colonnes sont les vecteurs $1, t, \overline{t^2}, ..., \overline{t^{D-1}}$ et B la matrice dont les colonnes sont $\overline{t^D}$, $\overline{X_1}$, ..., $\overline{X_n}$, c'est à dire une base de Gröbner lexicographique de l'idéal $\mathcal{I} + < T - t >$ pour n'importe quel ordre lexicographique tel que $T < X_i, i = 1...n$ par une version simplifiée de l'algorithme FGLM [47] (l'algorithme PREDICTSEPELT a été utilisé pour calculer t et a détecté que $\mathcal{I}$ est *shape position*, alors M est inversible).

On obtient ainsi une RUR de la forme $p_t(T)$, $X_1 - p_{t,X_1}(T)$, ..., $X_n - p_{t,X_n}(T)$ ou les coefficients de p_t (resp. p_{t,X_i}) sont les scalaires de la première (resp. $i+1$-ième) colonne de X.

Si on observe la complexité d'un tel processus, les matrices M et B se construisent en $O(D^3)$ opérations arithmétiques et les scalaires de la matrice M sont de taille $O(D(s + D))$. La résolution de $M\,X = B$ nécessitera alors $O(D^2(D + n))$ opérations arithmétiques avec des coefficients de taille au plus $O(D^2(s + D))$ ce qui est plus coûteux que le calcul d'un candidat-RUR dans le cas général à l'aide de la table de multiplication.

Notons $\{f_t(T),\ g_{t,1}(T),\ g_{t,X_1}(T),\ ...,\ g_{t,X_n}(T)\}$ le candidat-RUR que nous aurions obtenus pour le même élément séparant t par le corollaire 3.16, autrement dit par n'importe quelle variante de l'algorithme RUR-CANDIDATE. Il y a un lien direct entre la base lexicographique ci-dessus et ce candidat-RUR :

Lemme 3.24. ***FR**-[113], **FR**-[114] Soient $I \subset \mathbb{Q}[X_1, ..., X_n]$ un idéal zéro-dimensionnel et $t \in \mathbb{Q}[X_1, ..., X_n]$ tels que $I + < T - t > \subset \mathbb{Q}[T, X_1, ..., X_n]$ soit* shape position *pour T. Si $\{p_t(T), X_1 - p_{t,X_1}(T), ..., X_n - p_{t,X_n}(T)\}$ est une base de Gröbner de $I + < T - t >$ pour une ordre lexicographique tel que $T < X_i$, $i = 1...n$ et que $\{f_t(T),\ g_{t,1}(T),\ g_{t,X_1}(T),\ ...,\ g_{t,X_n}(T)\}$ est un candidat-RUR alors :*

- $f_t = p_t$;

- $g_{t,1}(T)$ *est inversible modulo* f_t*;*
- $p_{t,X_n}(T) = (g_{t,1}(T))^{-1} g_{t,X_n}(T) \bmod f_t$

Comme les coefficients du candidat-RUR sont de taille $O(D(s+D))$, il semble préférable, en tout cas du point de vue de la complexité, de calculer ce type d'objet plutôt qu'une base de Gröbner pour un ordre lexicographique, comme le confirment les expériences menées sur quelques exemples dans **FR**-[114] :

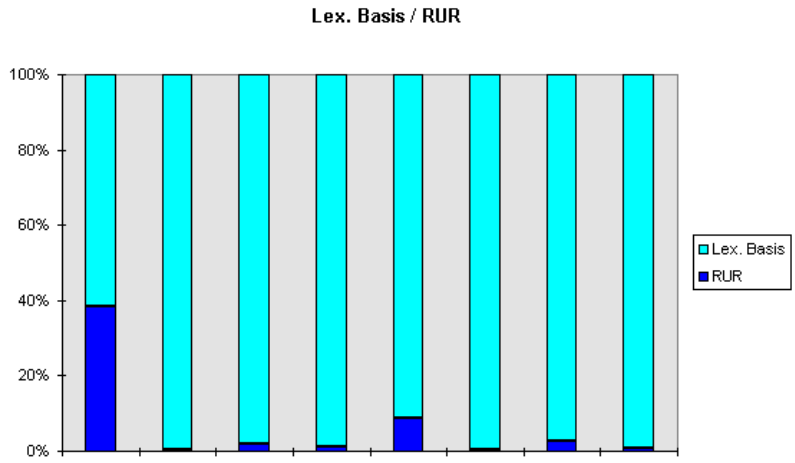


Figure 3.2. Taille des coefficients base lex/RUR

Remarquons que le lemme précédent suggère immédiatement de remplacer B par B', la matrice dont les colonnes sont $\overline{t^D}$, $\overline{g_{t,1}(t)X_1}$, ..., $\overline{g_{t,1}(T)X_n}$ dans l'algorithme de type FGLM vu plus haut. La résolution du système $M\,X = B'$ donne alors directement les coefficients du candidat-RUR. En pratique, on ne peut construire entièrement B tant que l'on ne connaît pas f_t, aussi il convient de séparer un peu plus les calculs :

Algorithm RUR-ShapePosition

- **Input :** $M_{X_i}, t, \mathcal{B}$
- **Output :** a RUR $\{f_t(T), g_{t,1}(T), g_{t,X_1}(T), ..., g_{t,X_n}(T)\}$
- for $i = 2...D$ $\overline{t^i} = M_t \overline{t^{i-1}}$
- Compute M^{-1} where M is the matrix whose columns are $1, t, \overline{t^2}, ..., \overline{t^{D-1}}$
- Compute $X = M^{-1}\,\overline{t^D}$ and set $f_t = T^D - \sum_{i=1}^{D} X[i]\,T^{i-1}$
- Compute $g_{t,1}(T) = \frac{\partial f_t}{\partial T} = \sum_{i=0}^{D-1} a_i T^i$
- Compute $\overline{g_{t,1}(t)} = \sum_{i=0}^{D-1} a_i \overline{t^i}$
- for i=1..n
  - $\overline{g_{t,1}(t)X_i} = M_{X_i}\overline{g_{t,1}(t)}$
  - Compute $X = M^{-1}\overline{g_{t,1}(t)X_i}$ and set $g_{t,X_i} = \sum_{i=1}^{D} X[i]\,T^{i-1}$
- Compute $h(T) = \gcd(f_t, \frac{\partial f_t}{\partial T})$
- for $v \in \{1, X_1, ..., X_n\}$ set $g_{t,v}(T) = \frac{g_{t,v}(T)}{h(T)}$

Remarque 3.25. Les 2 dernières étapes de calcul peuvent être omises : dans ce cas, les fractions rationnelles $\frac{g_{t,X_i}(T)}{g_{t,1}(T)}$ ne seront pas réduites et en particulier, le dénominateur pourra s'annuler en certains zéros de f_t.

Même si on assure une taille de coefficients en sortie qui soit plus petite en calculant une RUR au lieu d'une base lexicographique, il n'en demeure pas moins que les tailles de coefficients dans les calculs intermédiaires restent les mêmes dans les deux stratégies. En l'occurrence, ce qui pollue la complexité de l'algorithme est la taille des coefficients apparaissant dans M^{-1} puisqu'elle est en $O(D^2(s+D))$.

Un moyen simple et efficace d'éviter cette croissance intermédiaire est d'utiliser une méthode multi-modulaire ou p-adique comme dans [44] ou encore [101]. La différence entre les deux étant en pratique minime, nous nous attacherons qu'aux méthodes multi-modulaires, plus faciles à décrire.

Le principe théorique est simple : supposant que les coefficients du résultat sont des rationnels dont les numérateurs et dénominateurs sont majorés par une borne P, on suppose que l'on connaît un produit d'entiers premiers distincts $P^2 < \prod_{i=1}^{r} p_i$ tels que la matrice M mod p_i soit inversible pour tout $i = 1...r$ et que les dénominateurs apparaissant dans le résultat $X = [\frac{n_i}{d_i}, i = 1...D]$ sont tous inversibles modulo chacun des p_i, $i = 1...D$. On résout alors les systèmes $M_{p_i}\ X_{p_i} = B_{p_i}$ dans $\frac{\mathbb{Z}}{p_i\mathbb{Z}}$, ou M_{p_i} (resp. B_{P_i}) est l'image de M (resp. B) modulo p_i, puis on applique le théorème des restes Chinois pour reconstruire l'unique vecteur $X_P = [e_i, i = 1...D]$ à coefficients dans $\frac{\mathbb{Z}}{P\mathbb{Z}}$ dont l'image modulo p_i est X_{P_i}. Enfin on reconstruit l'unique vecteur $X = [\frac{n_i}{d_i}, i = 1...D]$ à coefficients rationnels tel que $e_i = d_i^{-1}\, n_i \bmod P$.

En pratique, il n'est pas concevable d'utiliser une borne théorique ni de garantir a priori les hypothèses sur les entiers premiers. On utilise alors les 2 remarques suivantes :

- il n'est pas vrai que si le procédé de reconstruction rationnelle donne un résultat, celui-ci soit le bon. Par contre, si deux remontées rationnelles consécutives (pour $\prod_{i=1}^{r'} p_i$ et $\prod_{i=1}^{r'+1} p_i$) sont égales, alors le résultat est probablement correct;
- un « mauvais » entier premier se détecte facilement (non inversibilité de M).

Détectant les « mauvais » entiers premiers et utilisant l'euristique ci-dessus, on a alors un algorithme probabiliste pour calculer un candidat-RUR qu'il faut maintenant certifier (remontée rationnelle + élément séparant).

Par construction, il faut que $f_t(t) \in \mathcal{I}$, sinon, la remontée rationnelle des coefficients de ce polynôme a échoué (pas assez d'entiers premiers). Une fois cette condition remplie, on est déjà assurés que $\mathcal{I}$ est *shape position* puisque la matrice M est nécessairement inversible, prouvant que le polynôme caractéristique de M_t est également son polynôme minimal.

Il faut encore s'assurer que les polynômes $g_{t,v}(T)$, $v \in \{1, X_1, ..., X_n\}$ ont été correctement reconstruits (que suffisamment d'entiers premiers ont été utilisés), ce qui revient simplement à vérifier que $g_{t,1}(t)\, h(t) X_i - g_{t,X_i}(t) h(t) \in \mathcal{I}, \forall i = 1...n$, $h(T) = \gcd(f_t, \frac{\partial f_t}{\partial T})$ (voir simplification effectuée dans l'algorithme RUR-SHAPEPOSITION).

Pour tout polynôme $P(T) = \sum_{i=0}^{D} a_i T^i \in \mathbb{Q}[T]$, le calcul de $P(t) \bmod \mathcal{I}$, ou de façon équivalente celui de $\overline{P(t)}$, peut se faire simplement en calculant les vecteurs $\overline{t^i} = M_t \overline{t^{i-1}}$ avec $\overline{t^0} = [1, 0, ..., 0]$, puis en les combinant $\overline{P(t)} = \sum_{i=0}^{D} a_i \overline{t^i}$. Il en coûte donc $O(D^4(D+s))$ opérations binaires pour vérifier si t est séparant et assurer que f_t a correctement été remonté, puis $O(n\, D^3(D+s))$ opérations binaires pour calculer $\overline{g_{t,v}(t)\, h(t)}$. Il ne reste plus qu'à vérifier que les vecteurs $M_{X_i}\ \overline{g_{t,1}(t)\ h(t)} - \overline{g_{t,X_i}(t) h(t)}$ sont tous nuls pour garantir que les polynômes $g_{t,v}(T)$, $v = 1, X_1, ..., X_n$ ont été correctement remontés soit $O(n\, D^3(D+s))$ opérations binaires.

Remarque 3.26. Dans tous les cas, on observe que le calcul d'un candidat-RUR dans le cas d'un idéal *shape position* est toujours plus facile que dans le cas général.

Même si une méthode p-adique peut éventuellement donner un meilleur résultat, nous retiendrons de cette partie les éléments suivants :

Comp.	binary op.	arith. Op.
RUR-SHAPEPOSITION (MULTI-MODULAR)	$O(D^4(s+D))$	$O(\ D^3 + n\, D^2)$
CHECK RUR-SHAPEPOSITION	$O\left((D^4 + n\, D^3)(D+s)\right.$	$O(D^3 + n\, D^2)$

En particulier, il est définitivement plus facile de calculer une RUR dans le cas d'un idéal *shape position*.

3.5.3 L'algorithme général

En mettant bout à bout l'ensemble des fonctions étudiées dans ce chapitre, on peut proposer un algorithme général efficace « en moyenne » pour le calcul de RUR, même dans le cas particulier d'idéaux *shape position* :

Algorithm RUR

- **Input :** $M_{X_i}\mathcal{B}$
- **Output :** a RUR $\{f_t(T), g_{t,1}(T), g_{t,X_1}(T), ..., g_{t,X_n}(T)\}$
- (*) choose a large prime number p
- t, IsShapePosition := ()
- if IsShapePosition then
  - $R :=$ RUR-ShapePosition[MultiModular]$(M_{X_i}, \mathcal{B}, t)$
  - If (CheckRUR-ShapePosition()=FAIL) then goto (*)
- else
  - $R :=$ RURCandidate$(M_{X_i}, \mathcal{B}, t)$
  - If (CheckRUR$(R, M_{X_i}, \mathcal{B}, t)$=FAIL) then goto (*)
- return(R)

La complexité de cet algorithme est, en moyenne (c'est à dire si l'on veut bien admettre que les « mauvais » entiers premiers sont rares) en $O(D^3 + n\ D^2)$ opérations arithmétiques dans le cas d'un idéal *shape position générique* et en $O(\#\mathcal{T}D^2 + n\ D^{5/2})$ dans le cas général si on utilise une stratégie de type « pas de bébé / pas de géant », $O(\#\mathcal{T}\ D^2 + n\ D^2)$ sinon. En termes de complexité binaire, ceci donne $O((D^4 + n\ D^3)(D + s))$ dans le cas d'un idéal *shape position générique* et $O((\#\mathcal{T}D^2\delta + n\ D^{7/2})(s + D))$ dans le cas général si on utilise une stratégie de type « pas de bébé / pas de géant », $O((\#\mathcal{T}D^2\delta + D^4 + n\ D^3)(D + s))$ sinon.

3.6 Extension aux systèmes avec inégalités/inéquations

Dans le cas de systèmes d'équations zéro-dimensionnels, nous avons vu que la RUR fournit un système « équivalent » au système de départ, permettant de ramener systématiquement le problème à l'étude de polynômes en une variable. Dans beaucoup d'applications, on désire extraire les solutions vérifiant des inéquations ou des inégalités (par exemple caractériser les zéros qui n'ont que des coordonnées positives).

Supposons que l'on se donne un système d'équations $\mathcal{S} = \{p_1(X_1, ..., X_n) = 0, ..., p_r(X_1, ..., X_n) = 0\}$ et que l'on désire en étudier les solutions telles que $q_1(X_1, ..., X_n) \neq 0$, ce qui est le cas le plus simple.

On pourrait imaginer calculer une RUR de $\mathcal{I} = \langle p_1, ..., p_s\rangle$ c'est à dire une expression formelle des solutions de la forme $\{f_t(T) = 0, X_i = \frac{g_{t,X_i}(T)}{g_{t,1}(T)}, i = 1...n\}$, calculer $q_1(\frac{g_{t,X_n}(T)}{g_{t,1}(T)}, ..., \frac{g_{t,X_n}(T)}{g_{t,1}(T)})$ ou plutôt $Q(T) = g_{t,1}(T)^d q_1(\frac{g_{t,X_n}(T)}{g_{t,1}(T)}, ..., \frac{g_{t,X_n}(T)}{g_{t,1}(T)})$, d étant le degré total de q_1, pour se débarrasser des dénominateurs (rappelons que $g_{t,1}$ ne s'annule pas en les zéros de f_t), puis calculer $G(T) = \gcd(\text{squarefreepart}(Q(T)), \text{squarefreepart}(f_t(T)))$ et $R(T) = \frac{\text{squarfreepart}(f_t(T))}{G(T)}$: les zéros de $\mathcal{I}$ en lesquels q_1 ne s'annule pas pourront être alors définis par $\{R(T) = 0, X_i = \frac{g_{t,X_i}(T)}{g_{t,1}(T)}, i = 1...n\}$, ceux pour lesquels q_1 s'annule pourront l'être par $\{G(T) = 0, X_i = \frac{g_{t,X_i}(T)}{g_{t,1}(T)}, i = 1...n\}$.

Cette méthode est bien sûr naïve car le calcul de $Q(T)$ est extrêmement difficile à effectuer. Même en imaginant que Q soit réduit modulo $\mathcal{I}$, c'est à dire qu'il s'exprime comme combinaison linéaire d'éléments de la base $\mathcal{B}$ de $\frac{\mathbb{Q}[X_1,...,X_n]}{\mathcal{I}}$, il faudra au moins calculer $g_{t,1}(T)^D$ modulo f_t pour obtenir le résultat. Quelle que soit la méthode employée, le coût binaire de cette opération sera au moins égal à celui du calcul complet de la RUR.

Plus généralement, on suppose que l'on désire étudier les polynômes $q_1(X_1, ..., X_n)$, ..., $q_l(X_1, ..., X_n)$ en les racines de $\mathcal{S}$. Une solution possible, plus acceptable que la précédente, consiste à étudier les zéros de l'idéal $\langle p_1, ..., p_r, T_1 - q_1, ..., T_l - q_l\rangle \subset \mathbb{Q}[T_1, ..., T_l, X_1, ..., X_n]$, par exemple en en calculant une RUR $\{f_t(T) = 0, X_i = \frac{g_{t,X_i}(T)}{g_{t,1}(T)}, i = 1...n, T_i = \frac{g_{t,T_i}(T)}{g_{t,1}(T)}, i = 1...l\}$: il sera alors facile de tester la nullité des coordonnées ainsi que celle des polynômes q_i, en calculant simplement les pgcds des g_{t,T_i} ou g_{t,X_i} et de f_t. L'inconvénient de cette stratégie est de devoir systématiquement dérouler toute la chaîne de calcul, y compris la représentation de l'algèbre quotient (prérequis pour calculer une RUR) à chaque étude d'un nouveau polynôme q_i, ceci étant structurellement inutile puisque l'on peut remarquer que $\frac{\mathbb{Q}[X_1,...,X_n]}{\langle p_1,...,p_r\rangle}$ est isomorphe à $\frac{\mathbb{Q}[T_1,...,T_l,X_1,...,X_n]}{\langle p_1,...,p_r,T_1-q_1,...,T_{l-q_l}\rangle}$.

En remarquant que les formules du corollaire 3.16 permettent de définir $g_{t,v}$ pour tout polynôme $v \in \mathbb{Q}[X_1, ..., X_n]$ et en supposant que les q_i soient déjà réduits modulo $\mathcal{I}$ (au besoin on peut les réduire en utilisant les prérequis donnés par la Notation 3.4), on peut remarquer **FR**-[117] que $g_{t,q_i} = g_{t,T_i}$ et que le calcul de ces fonctions est de même complexité que celui des g_{t,X_i}. Ceci peut se résumer par :

Lemme 3.27. ***FR**-[117] Soit $\mathcal{I} \subset \mathbb{Q}[X_1, ..., X_n]$ un idéal zéro-dimensionnel et $q_1, ..., q_l$ des polynômes de $\mathbb{Q}[X_1, ..., X_n]$ réduits modulo $\mathcal{I}$ (plus précisément s'exprimant comme combinaisons linéaires d'éléments de $\mathcal{B}$ si l'on utilise la notation 3.4). On appelle RUR de $\mathcal{I}$ étendue aux polynômes $q_i, i = 1...l$, l'ensemble $\{f_t(T), g_{t,1}(T), g_{t,X_1}(T), ..., g_{t,X_n}(T), g_{t,q_1}(T), ..., g_{t,q_l}(T)\}$ tel que*

- *$\{f_t(T), g_{t,1}(T), g_{t,X_1}(T), ..., g_{t,X_n}(T)\}$ est une RUR de $\mathcal{I}$;*
- *$q_i(\alpha) = \frac{g_{t,q_i}(t(\alpha))}{g_{t,1}(t(\alpha))}, \forall i = 1...l, \forall \alpha \in \boldsymbol{V}(\mathcal{I})$*

Les polynômes g_{t,q_i} se calculant par la formule générale du corollaire 3.16, on peut simplement dériver n'importe quelle variante de l'algorithme RUR *présenté à la section précédente pour calculer une RUR-étendue en $O(R + l\,D^2)$ opérations arithmétiques ou encore $O(\mathrm{Rbin} + n\,D^3(D + s))$ opérations binaires, $O(R)$ (resp. $O(\mathrm{Rbin})$) étant le nombre d'opérations arithmétiques (resp. binaires) nécessaires pour calculer une RUR de $\mathcal{I}$.*

Nous verrons dans la section qui suit que le calcul d'une RUR de $\mathcal{I}$ étendue à une famille de polynômes q_1, ..., q_l permet d'évaluer efficacement les q_i en les zéros d'un système, mais le résultat précédent à des conséquences immédiates sur la complexité des algorithmes permettant simplement d'étudier le signe des q_i en les zéros de $\mathcal{I}$. En effet, nous avons vu dans le chapitre 2 un moyen efficace d'évaluer les signes d'un ensemble de polynômes en une variable en les zéros d'un autre polynôme en une variable. Si l'on applique cette fonction pour étudier les signes des g_{t,q_i} et de $g_{t,1}$ en les zéros de f_t, on obtient alors instantanément un algorithme donnant les signes des q_i en les zéros de $\mathcal{I}$, améliorant d'un facteur au moins D l'algorithme « inégalités simultanées » [17],[122],[13] réalisant le même calcul par le biais du théorème 3.10.

3.7 Résolution certifiée de systèmes zéro-dimensionnels

La RUR fournit un système « équivalent » au système initial mais permet surtout de se ramener à la résolution de problèmes ne dépendant plus que d'une variable. Par exemple, si l'on désire compter le nombre de solutions réelles d'un système, c'est à dire le nombre de points de $\boldsymbol{V}(\mathcal{I}) \cap \mathbb{R}^n$, il suffit alors de compter le nombre de solutions réelles de $\boldsymbol{V}(f_t)$, par exemple en utilisant un des algorithmes présentés dans le chapitre 2.

Dans la plupart des applications, résoudre un système (zéro-dimensionnel) sous-entend calculer une approximation de ses solutions. Mathématiquement, on peut estimer le problème résolu lorsque l'on connaît une RUR puisqu'il n'y a « plus qu'à » calculer les racines de f_t puis à les « substituer » à T dans les fractions rationnelles $\frac{g_{t,X_i}(T)}{g_{t,1}(T)}$. Compte tenu des objectifs de ce document, nous nous concentrerons sur les solutions réelles.

En pratique, les choses sont plus compliquées puisque le codage d'un nombre réel, même algébrique, est non trivial en pratique. Il existe de nombreuses façons de représenter un nombre réel algébrique : certaines sont définitivement approximatives (nombres flottants), d'autres exactes mais limitées dans leur utilisation (intervalles a bornes rationnelles, codage à la Thom), d'autres formelles et coûteuses à utiliser (développement en série formelle au voisinage d'un point).

Un choix naturel est de représenter un nombre réel algébrique α par un polynôme f sans facteurs carrés tel que $f(\alpha) = 0$ et un intervalle à bornes rationnelles contenant α et aucune autre racine de f, simplement parce que nous disposons d'un moyen efficace de calculer de telles représentations (voir chapitre 2) mais aussi parce que ce type de codage n'est pas limitant (il permet de raffiner simplement l'intervalle d'isolation et donc d'obtenir une approximation numérique avec une précision arbitraire mais également d'effectuer efficacement quelques opérations simples comme comparer deux nombres réels algébriques).

En première approximation, on peut :

- isoler les racines de f_t et raffiner les intervalles (il suffit de procéder par dichotomie en évaluant f_t aux bornes des intervalles d'isolation);
- utiliser une arithmétique d'intervalles, éventuellement multi-précision (comme décrite dans le chapitre 2) pour évaluer les polynômes $g_{t,v}$, $v = 1$, X_i, $i = 1...n$ (et éventuellement les polynômes g_{t,q_i}, $i = 1...l$ dans le cas d'une RUR étendue).

Supposons qu'une racine α de f_t soit représentée par un intervalle en précision p $\lfloor \alpha \rceil_p$ (voir notations de la section 2.7 du chapitre 2). Rappelons qu'utiliser de l'arithmétique d'intervalle en précision fixe pour évaluer un polynôme Q en α permet de calculer un intervalle $\lfloor \beta \rceil_p$ contenant toutes les valeurs $Q(\gamma)$, $\gamma \in \lfloor \alpha \rceil_p$, mais que cette inclusion peut être stricte (erreur d'approximation).

Utiliser ce type de stratégie donnera des résultats acceptables pour des systèmes de difficulté moyenne mais elle se heurte en général à plusieurs problèmes structurels :

- [A] : elle ne permet pas de décider du signe d'une coordonnée (ou d'un polynôme dans le cas d'une RUR étendue), sauf dans certains cas très particuliers. L'accumulation d'erreurs d'approximation peut engendrer l'apparition de 0 dans l'intervalle contenant le résultat;
- [B] : multiplier des intervalles dont les bornes sont représentées par des nombres flottants en précision fixe par des rationnels de grande taille (par exemple les coefficients des polynômes d'une RUR) engendre des erreurs importantes (largeur des intervalles codant le résultat de telles opérations);
- [C] l'évaluation d'un polynôme de degré D avec une arithmétique d'intervalles engendre une erreur numérique (croissance artificielle de l'intervalle codant le résultat) de l'ordre de $O(D)$.

Le problème [A] se règle facilement en calculant le pgcd des polynômes $g_{t,v}$, $v = X_1, ..., X_n$ (et éventuellement $v = q_1, ..., q_l$ dans le cas d'une RUR étendue) et de f_t permettant ainsi de discriminer les racines de f_t en fonction de la nullité des coordonnées.

Le problème [C] peut se régler en choisissant une précision pour l'arithmétique d'intervalle d'au moins $k\,D$ bits pour la mantisse (k étant un entier à déterminer en fonction de la qualité de l'arithmétique utilisée), mais le problème [B] est plus difficile à appréhender en pratique. En effet, la largeur des intervalles du résultat final dépend de l'erreur commise (problèmes [B] et [C]) dans l'évaluation des fractions rationnelles $\frac{g_{t,X_i}(T)}{g_{t,1}(T)}$ en un intervalle, mais également de la largeur de cet intervalle. Dès lors que la précision du résultat n'est pas celle escomptée, ceci force à raffiner l'intervalle d'isolation puis à ré-évaluer la fraction rationnelle avec une précision supérieure pour l'arithmétique.

En pratique, l'implantation de ce procédé a mis en évidence que le raffinement d'un intervalle d'isolation par dichotomie (évaluation de f_t en les deux bornes de l'intervalle ainsi qu'en un point de l'intervalle) s'avère extrêmement coûteux dès lors que l'intervalle d'isolation est de faible largeur. Pour donner une ordre de grandeur, raffiner les intervalles d'isolation des polynômes orthogonaux utilisés pour les tests du chapitre 2 pour obtenir une largeur $< 1/2^{32}$ s'avère aussi coûteux que le calcul de ces intervalles.

Le réflexe est alors d'utiliser l'algorithme de Newton dont la convergence est plus rapide pour raffiner les intervalles d'isolation. La contrainte est d'en garantir son comportement, c'est à dire au moins s'assurer que $\frac{\partial f_t}{\partial T}$ ne s'annule pas sur l'intervalle considéré. Pour ce faire, on peut utiliser l'algorithme du chapitre 2 et, quitte à utiliser ce genre de stratégie, on peut également étudier les dérivées des autres polynômes constituant la RUR. Au final, on aura raffiné les racines de f_t de sorte à assurer que les polynômes $g_{t,v}$, $v = 1$, X_1, ..., X_n sont monotones au dessus des intervalles d'isolation des racines de f_t (voir l'algorithme du chapitre 2). L'intérêt majeur est que si l'on cherche à connaître les valeurs prises par ces polynômes lorsque leur variable parcourt un intervalle d'isolation de f_t, il suffit de les évaluer aux bornes de cet intervalle.

3.8 Conclusion

Les résultats présentés dans ce chapitre permettent de construire la « boite noire » suivante, qui généralise celle proposée en conclusion du chapitre 2 :

SOLVE-ZERODIM

- **Input** : $\{p_1, ..., p_r\} \in \mathbb{Q}[X_1, ..., X_n]$, $\mathcal{F} = \{q_1, ..., q_l\}$, $q_i \in \mathbb{Q}[X_1, ..., X_n]$, p an integer,
  - PREPROC, a function which tests if $\langle p_1, ..., p_r \rangle$ is zero-dimensional or not and, if so, computes $\mathcal{B}$, a monomial basis of $\frac{\mathbb{Q}[X_1, ..., X_n]}{\langle p_1, ..., p_r \rangle}$;
  - REDUCTION, a function which computes P mod $\mathcal{I}$ and express the result with respect to $\mathcal{B}$ for any $P \in \mathbb{Q}[X_1, ..., X_n]$;
  - SOLVE-UNIVARIATE
- **Output** :
  - FAIL if $\#\boldsymbol{V}(\langle p_1, ..., p_s \rangle) = \infty$
  - An extended RUR $\{f_t, X_i = \frac{g_{t,X_i}(T)}{g_{t,1}(T)}, i = 1...n, q_i = \frac{g_{t,q_i}(T)}{g_{t,1}(T)}, i = 1...l\}$
  - $\{\prod_{k=1}^{n} (a_{i,k}^j, b_{i,k}^j), i = 1...d, j = 0...l\}$ with $a_{i,k}^j, b_{i,k}^j \in \mathbb{Q}$, such that
    - $(a_{i,k}^j, b_{i,k}^j) = [a_{i,k}^j, a_{i,k}^j] = \{a_{i,k}^j\}$ if $a_{i,k}^j = b_{i,k}^j$ and $(a_{i,k}^j, b_{i,k}^j) =]a_{i,k}^j, b_{i,k}^j]$ if $a_{i,k}^j \neq b_{i,k}^j$. Moreover, $a_{i,k}^j$ and $b_{i,k}^j$ have the same sign;
    - $\alpha \in \boldsymbol{V}(\langle p_1, ..., p_s \rangle) \cap \mathbb{R}^n$ iff $\exists i, \alpha \in \prod_{k=1}^{n} (a_{i,k}^0, b_{i,k}^0)$
    - $\prod_{k=1}^{n} (a_{i,k}^0, b_{i,k}^0)$ contains a unique element of $\boldsymbol{V}(\langle p_1, ..., p_s \rangle) \cap \mathbb{R}^n$
    - if $\alpha \in \boldsymbol{V}(\langle p_1, ..., p_s \rangle) \cap \mathbb{R}^n$ then $\exists i$, $\alpha \in \prod_{k=1}^{n} (a_{i,k}^0, b_{i,k}^0)$ and $q_j(\alpha) \in \prod_{k=1}^{n} (a_{i,k}^j, b_{i,k}^j), \forall j = 1...l$
    - $|a_{i,k}^j - b_{i,k}^j| < 1/2^p, \forall i = 1...d, j = 1...l, k = 1...n$
  - a function REFINE taking as input $\{f_t, X_i = \frac{g_{t,X_i}(T)}{g_{t,1}(T)}, i = 1...n, q_i = \frac{g_{t,q_i}(T)}{g_{t,1}(T)}, i = 1...l\}$, $\{\prod_{k=1}^{n} (a_{i,k}^j, b_{i,k}^j), i = 1...d, j = 1...l\}$ and an integer $p' > p$, which returns $\{\prod_{k=1}^{n} (c_{i,k}^j, d_{i,k}^j), i = 1...d, j = 1...l\}$ with the same properties as $\{\prod_{k=1}^{n} (a_{i,k}^j, b_{i,k}^j), i = 1...d, j = 0...l\}$ but such that $|c_{i,k}^j - d_{i,k}^j| < 1/2^{p'}, \forall i = 1...d, j = 0...l, k = 1...n$.

On a pu voir dans ce chapitre que, selon le but recherché (complexité arithmétique, complexité binaire, certification, efficacité en moyenne, etc...), les stratégies (choix/vérification d'un élément séparant) ainsi que les outils de calcul (RUR-Candidate) diffèrent très sensiblement. On a pu voir aussi que la limitation à un cas particulier (idéaux *shape position*) permet d'optimiser beaucoup les algorithmes pourvu que l'on sache détecter facilement ce type de situation.

Nous avons montré que 2 aspects souvent négligés : l'obtention d'un élément séparant et la gestion des multiplicités, sont, d'un point de vue algorithmique, les points les plus délicats à traiter pour la résolution symbolique (élimination de variables) des systèmes zéro-dimensionnels. En particulier comparer un algorithme probabiliste tel que [56] ou ne résolvant qu'un cas particulier comme [101] (en général les alternatives proposées supposent l'idéal radical) avec la méthode générale n'a donc pas vraiment de sens. Toutefois, si l'on fait abstraction de ces éléments, l'algorithme proposé dans ce chapitre se compare favorablement aux méthodes formelles existantes ainsi qu'aux techniques numériques les plus éprouvées.

Entre la version préliminaire **FR**-[113] et les versions les plus abouties (utilisation d'algorithmes basées sur la règle de Descartes, utilisation éventuelle de « pas de bébé/pas de géant », de calcul multi-modulaire, etc.) on peut noter un progrès en complexité théorique que ce soit sur le nombre d'opérations arithmétiques ou sur le nombre d'opérations binaires mais aussi un net progrès en pratique. Par exemple, alors qu'il fallait plusieurs jours de calcul pour simplement compter les solutions de certains modèles géométriques directs pour les robots parallèles, les temps de calculs actuels sont de l'ordre de la seconde pour obtenir une approximation certifiée des solutions (donc en particulier leur nombre) pour n'importe quel robot.

Mais le progrès essentiel porte certainement sur les classes d'exemples que l'on peut désormais appréhender : on peut désormais dire que la résolution d'un système admettant quelques centaines de solutions complexes (comptées avec multiplicités) est un problème facile (on parle en secondes/minutes de calcul); en général, un système admettant jusqu'à environ un millier de racines ne résiste que rarement (on parle de minutes/heures de calcul) aux « boites noires » proposées. Au delà d'un millier de solutions, il faut être un peu plus chanceux : il est, en particulier, fortement souhaitable d'avoir à faire à un système radical ou pour le moins *shape position* de sorte à utiliser la stratégie multi-modulaire (ce qui permet en particulier de distribuer les calculs). La barre est désormais placée à la dizaine de milliers de solutions, barre déjà atteignable dans certains cas particuliers. Pour ces exemples « extrêmes », les problèmes ne sont plus les mêmes : le facteur bloquant est, le plus souvent, l'occupation mémoire.

Chapitre 4
Systèmes de dimension positive

4.1 Introduction

Le calcul d'un point par composante connexe d'un semi-algébrique est un challenge de longue date en géométrie algébrique réelle effective. En théorie, il représente un outil de base pour l'étude des problèmes d'élimination de quantificateurs, que ce soit avec le point de vue de la Décomposition Cylindrique Algébrique [30] ou pour les stratégies plus récentes comme celles proposées dans [11], [14] [9], que l'on retrouve en détail dans [13].

En termes d'applications, ce problème appariait souvent de façon indirecte, soit comme sous problème pour un autre algorithme (voir chapitre 5), soit comme outil permettant de tester si un système général admet ou non des solutions réelles (voir applications à la synthèse de bancs de filtres ou le problème d'interpolation de Birkhoff dans le chapitre 6) ou simplement pour calculer des solutions particulières.

Deux stratégies générales s'opposent. Le problème peut être vu comme résolu par un algorithme de Décomposition Cylindrique Algébrique, puisque ce procédé fournit une partition de l'espace ambiant en cellules (homéomorphes à des boules) dans lesquelles des polynômes donnés gardent un signe constant. La seconde catégorie principale rassemble des méthodes qui sont essentiellement basées sur la détermination d'une fonction atteignant ses extrema, en nombre fini, sur chaque composante connexe de l'ensemble étudié (on peut par exemple citer [129] qui est peut-être l'algorithme le plus ancien). Mettre en équations ces extrema permet alors de ramener le problème à l'étude de systèmes zéro-dimensionnels.

Bien qu'il ait été montré que les stratégies basées sur une CAD sont systématiquement doublement exponentielles en le nombre de variables, alors que certaines méthodes de points critiques sont simplement exponentielles [11], [14] [9], le caractère praticable de ces dernières n'a été mis en évidence que très tardivement, en particulier grâce aux contributions présentées dans ce chapitre.

Pour mesurer la pression sur le sujet, on peut citer un rapport de recherche de Hoon Hong datant de 1991 [68] qui, sur la base d'exemples précis conclut :

for small inputs, Critical Points Methods running time would be greater than 1 million years while CAD runs in less than 2 sec.

Ce à quoi il a été opposé dans [67] :

For computing one point on each semi-algebraically connected component of a real algebraic set, CAD's output is doubly exponential in the number of varaibles while critical points method is single exponential.

Ce chapitre ne s'intéresse qu'aux cas d'ensembles algébriques (réels) et il n'a pour but que de montrer la démarche qui a conduit à hisser les méthodes de points critiques parmi les alternatives possibles pour le calcul d'un point par composante connexe. D'autres développements, s'appuyant en partie sur les stratégies employées dans ce document, ont été réalisés depuis, améliorant sensiblement les performances [125],[126] autant en pratique qu'en théorie. Aussi, le niveau de détail dans la description des stratégies de calcul sera moindre que dans les autres chapitres de ce document.

Si l'on suit au pied de la lettre la stratégie proposée dans [13], on ne peut que reconnaître le bien fondé du rapport [68], malgré les adaptations proposées dans [67]. En effet, l'algorithme proposé dans [13] pour le calcul d'un point par composante connexe d'un ensemble algébrique réel y est décomposé en plusieurs étapes :

- Supposant que le système étudié soit $\mathcal{S} = 0$ avec $\mathcal{S} = \{p_1, ..., p_r\}$, les p_i étant des polynômes de $\mathbb{Q}[X]$, on étudie $Q = p_1^2 + ... + p_r^2 = 0$ qui, d'un point de vu réel est un problème strictement identique.
- La fonction utilisée pour le calcul des points critiques est la fonction coordonnée x_1 : dès lors que $\boldsymbol{V}(Q) \cap \mathbb{R}^n$ est lisse et bornée, l'ensemble de ses points critiques est borné et intersecte chaque composante connexe de $\boldsymbol{V}(Q) \cap \mathbb{R}^n$.
- On force les hypothèses en déformant l'équation étudiée :
  - $Q_\Omega = Q^2 + (X_1^2 + ... + X_n^2 + X_{n+1}^2 - \Omega^2)$ ou X_{n+1} est une nouvelle variable et $\frac{1}{\Omega}$ un infinitésimal;
  - $Q_{\Omega,\epsilon} = (1-\epsilon)\, Q_\Omega + \epsilon \left(X_1^{2(d_1+1)} + ... + X_n^{2(d_n+1)} + X_{n+1}^6 - 3\,(\Omega - 1)^{2(d_1+1)} \right)$, ϵ étant un infinitésimal, d_i étant le degré de Q_Ω en X_i.
- Les points critiques de la coordonnée x_1 en restriction à $\boldsymbol{V}(Q_{\Omega,\epsilon}) \cap \mathbb{R}^{n+1}$ sont contenus dans $\boldsymbol{V}(\mathcal{S}_{\Omega,\epsilon})$ avec $S_{\Omega,\epsilon} = \{Q_{\Omega,\epsilon}, \frac{\partial Q_{\Omega,\epsilon}}{\partial X_2}, ..., \frac{\partial Q_{\Omega\epsilon}}{\partial X_n}\}$ que l'on sait être zéro-dimensionnel par construction. Notons $\mathcal{C}_{\Omega,\epsilon}$ cet ensemble de $\mathbb{R}\langle\frac{1}{\Omega}, \epsilon\rangle^{n+1}$. Le résultat principal utilisé dans l'algorithme est que l'on obtient alors un point par composante connexe de $\boldsymbol{V}(\mathcal{S}) \cap \mathbb{R}^n$ en considérant les limites des points de $\mathcal{C}_{\Omega,\epsilon}$ lorsque $\frac{1}{\Omega} \to 0$ et $\epsilon \to 0$, ne retenant que celles qui sont bornées, puis en prenant leur projection dans $\mathbb{R}^n$.
- Une remarque importante est que $S_{\Omega,\epsilon}$ est une base de Gröbner pour l'ordre du Degré Lexicographique, si bien que sa résolution ne fait appel qu'à des techniques d'algèbre linéaire dont le comportement est bien contrôlé (voir chapitre 3). En particulier, on peut supposer que ses solutions dans $\mathbb{R}\langle\frac{1}{\Omega}, \epsilon\rangle^{n+1}$ soient représentées par une Représentation Univariée Rationnelle, c'est à dire sous la forme $\{f_t(T) = 0, X_i = \frac{g_{X_i,t}(T)}{g_{1,t}(T)}, i = 1...n+1\}$, f_t, $g_{v,t}$, $v = 1$, X_1, $...X_{n+1}$ étant des polynômes $\mathbb{R}\langle\frac{1}{\Omega}, \epsilon\rangle[T]$. Une telle représentation permet alors d'obtenir un développement en série de toutes les coordonnées de toutes les solutions dont on pourra étudier les limites quand $\frac{1}{\Omega} \to 0$ et $\epsilon \to 0$ pour obtenir le résultat.

Dans [116], on montre qu'une telle stratégie n'est pas viable en pratique. Il est par exemple montré que le cas d'un système de 3 polynômes en 3 variables et de degré 2 en chacune des variables nécessite, pour calculer une RUR de l'idéal définissant les points recherchés, au moins l'inversion d'une matrice de degré > 4000 dont les coefficients sont des polynômes en 2 variables (Ω et ϵ), de degré au moins 10 en Ω.

Les sources de blocage sont diverses :

- Les élévations de degrés multiples c'est à dire les déformations nécessaires pour
  - se ramener au cas d'un unique polynôme;
  - rendre la variété compacte;
  - rendre la variété lisse;
- L'utilisation d'une arithmétique complexe et coûteuse en pratique (séries de Puiseux ou pour le moins polynômes en $\frac{1}{\Omega}$ et ϵ).

La contrainte d'avoir une variété compacte peut être contournée en remplaçant, par exemple, la fonction coordonnée x_1 par la fonction « distance à un point A bien choisi », notée $d(A, .)$ par la suite. En effet, si A n'est pas contenu dans la variété à étudier et, si il est choisi de sorte à ce que la fonction $d(A, .)$ admette un nombre fini de points critiques, alors l'ensemble de ces points intersecte chaque composante connexe de la variété étudiée (voir, par exemple, [129]). Nous nous en tiendrons à ce choix de fonction dans ce chapitre, pointant tout de même que les méthodes les plus récentes utilisent une fonction de projection selon une coordonnée [125],[126].

4.1.1 Le cas des variétés définies par une unique équation.

Dans cette section, on ignore la première phase du calcul et les élévations de degrés induites pour se concentrer le le cas d'une variété définie par une unique équation. L'objectif est donc principalement de ne pas effectuer systématiquement des déformations infinitésimales et donc d'alléger le coût des opérations de base. On ne donne ici que les grandes lignes de l'algorithme décrit dans **FR**-[119] permettant de pointer les difficultés calculatoires : l'algorithme complet ne sera pas décrit puisqu'il sera avantageusement remplacé par la suite par celui de [7], proposé dans la section suivante.

Notation 4.1. *En supposant que $P \in \mathbb{Q}[X_1, ..., X_n]$ est irréductible et que $A = (a_1, ..., a_n)$ est un point de $\mathbb{R}^n$ à coordonnées rationnelles, on définit :*

- *Sauf mention contraire, ϵ sera un infinitésimal positif.*
- *$d(A, M)$ la distance d'un point M quelconque au point A.*
- *$\boldsymbol{V}(\mathcal{C}(A)) = \{M \in \mathbb{C}^n, P(M) = 0 \wedge \text{grad}_M(P)//\overrightarrow{\text{AM}}\}$, $\text{grad}_M(P)$ étant le vecteur gradient de P au point $M = [X_1, ..., X_n]$. Les points de $\boldsymbol{V}(\mathcal{C}(A))$ sont les zéros de l'idéal engendré par $\mathcal{C}(A) = \{P,\ P_{1,A},\ ...,\ P_{r,A}\}$ ou les $P_{i,A}$ sont les mineurs 2×2 de la matrice $\begin{pmatrix} \frac{\partial P}{\partial X_1} & X_1 - a_1 \\ \vdots & \vdots \\ \frac{\partial P}{\partial X_n} & X_n - a_n \end{pmatrix}$.*
- *On appellera par la suite* point singulier *de P, tout zéro de l'idéal engendré par $\mathcal{J} = \{P, \frac{\partial P}{\partial X_i}, i = 1...n\}$.*

L'ensemble $\boldsymbol{V}(\mathcal{C}(A))$ est l'union des points critiques de la fonction « distance au point A » et des zéros de l'idéal engendré par $\mathcal{J}$ et l'on a :

Lemme 4.2. ***FR**-[118] Toute composante semi-algébriquement connexe de $\boldsymbol{V}(P) \cap \mathbb{R}^n$ intersecte l'ensemble $\boldsymbol{V}(\mathcal{C}(A)) \cap \mathbb{R}^n$.*

Le fait que $\boldsymbol{V}(\mathcal{C}(A))$ puisse contenir d'autres points que les points critiques de la fonction « distance à A », amène a considérer plusieurs cas pour mener les calculs :

- Si $\boldsymbol{V}(\mathcal{C}(A)) = \emptyset$, alors $\boldsymbol{V}(P) \cap \mathbb{R}^n = \emptyset$;
- Si $\sharp\boldsymbol{V}(\mathcal{C}(A)) < 0$, $\mathcal{C}(A)$ est donc un système zéro-dimensionnel et on peut calculer $\boldsymbol{V}(\mathcal{C}(A)) \cap \mathbb{R}^n$ en utilisant, par exemple, une Représentation Univariée Rationnelle (voir l'algorithme proposé au chapitre 3).
- $\mathcal{C}(A)$ n'est pas zéro-dimensionnel, ce qui peut être la conséquence de 2 phénomènes
  - un mauvais choix de point A : on peut imaginer par exemple que $\boldsymbol{V}(P)$ contienne un cercle de centre A et donc une infinité de points équi-distants de A;
  - P admet une infinité de points singuliers, autrement dit, le système $\{M \in \mathbb{C}^n, P(M) = 0, \text{grad}_M(P) = 0\}$ est de dimension positive.

Ainsi, un algorithme raisonnable commencera par calculer la dimension de $\mathcal{C}(A)$, par exemple via un calcul de base de Gröbner, et conclura dès lors que $\mathcal{C}(A)$ admet au plus un nombre fini de solutions complexes.

Lorsque $\dim(\mathcal{J}) = 0$ et que $\dim(\mathcal{C}(A)) > 0$, le point A est équi-distant d'une infinité de points de $\boldsymbol{V}(P)$. On peut montrer que de tels points A sont sur un ensemble de mesure nulle de $\mathbb{C}^n$, plus précisément :

Lemme 4.3. ***FR**-[118] Si $\mathcal{H} = \{(M, \lambda) \in \mathbb{C}^{n+1} \mid P(M) = 0, \text{grad}_M(P) \neq 0\}$, et si $\text{Jac}(S)$ désigne le jacobien d'un système admettant autant d'équations que d'inconnues, alors, l'ensemble constructible*

$\mathcal{A} = \{B = [b_1, ..., b_n] \in \mathbb{C}^n \mid \mathcal{H} \cap \boldsymbol{V}(Q_1 + b_1, , ..., Q_n + b_n, \text{Jac}(P, Q_1 + b_1, ..., Q_n + b_n)) \neq 0\}$ avec $Q_i = \lambda \frac{\partial P}{\partial X_i} - X_i$ est contenu dans un sous-ensemble algébrique strict de $\mathbb{C}^n$ et, pour tout point $A \notin \mathcal{A}$, A ne peut pas être équidistant d'une infinité de points de $\mathcal{C}(A)$.

Si P est de degré d, alors il existe un point $A \in \{O, ..., d\}^n$ tels que $A \notin \mathcal{A}$

Si on détecte que $\dim(\mathcal{J}) = 0$, il suffit de choisir $A \in \{O,\ ...,\ d\}^n$ en s'assurant que $\dim(\mathcal{C}(A)) = 0$ pour se ramener à l'étude d'un système zéro-dimensionnel.

La réelle difficulté est donc le cas ou $\dim(\mathcal{J}) > 0$, c'est à dire le cas ou P admet une infinité de points singuliers complexes. Dans la suite, nous suivons **FR**-[118] et reprenons l'idée de la déformation infinitésimale.

Une conséquence directe du théorème de Sard et remarque de base pour ce qui suit est que ni $P - \epsilon$ ni $P + \epsilon$ n'a aucun point singulier dans $\mathbb{C}\langle\epsilon\rangle^n$, ϵ étant un infinitésimal. En utilisant le même type d'arguments que pour les algorithmes présentés dans [13], il est possible d'adapter la stratégie générale décrite en début de ce chapitre :

Lemme 4.4. ***FR**-[118] Si, pour tout point $A \in \mathbb{R}^n$, on note $\mathcal{C}_{-\epsilon}(A) = \{P + \epsilon, P_{1,A}, ..., P_{r,A}\}$, $\mathcal{C}_{+\epsilon}(A) = \{P - \epsilon, P_{1,A}, ..., P_{r,A}\}$, $V(\mathcal{C}_\epsilon(A)) = \boldsymbol{V}(\mathcal{C}_{+\epsilon}(A)) \cup \boldsymbol{V}(\mathcal{C}_{-\epsilon}(A))$ et $\lim_0(\mathcal{C}_\epsilon(A))$ les limites des points de $\boldsymbol{V}(\mathcal{C}_\epsilon(A))$ lorsque $\epsilon \to 0$, alors $\lim_0(\mathcal{C}_\epsilon(A)) \cap \mathbb{R}^n$ intersecte chaque composante semi-algébriquement connexe de $\boldsymbol{V}(P) \cap \mathbb{R}^n$.*

Le problème du choix du point A se pose exactement dans les mêmes termes que dans le cas « sans déformation » : il peut être choisi, pour les même raisons, sur une grille $\{0, ..., d\}^n$, d étant le degré de P, mais le critère de choix est cette fois simplement d'imposer que les idéaux $\langle\mathcal{C}_{+\epsilon}\rangle$ et $\langle\mathcal{C}_{-\epsilon}\rangle$ soient zéro-dimensionnels, puisque $P - \epsilon$ et $P + \epsilon$ n'ont aucun point singulier.

En rassemblant les résultats obtenus jusqu'ici, on peut voir que dès lors que l'on peut calculer dimension d'un idéal $I \subset \mathbb{Q}\langle\epsilon\rangle[X_1, ..., X_n]$ (ce qui inclus également le cas ou $I \subset \mathbb{Q}[X_1, ..., X_n]$), on peut toujours construire un système zéro-dimensionnel $\mathcal{S}_\epsilon \subset \mathbb{Q}\langle\epsilon\rangle[X_1, ..., X_n]$ tel que $\lim_0(\boldsymbol{V}(\mathcal{S}_\epsilon)) \cap \mathbb{R}^n$ soit vide ou alors intersecte toutes les composantes semi-algébriquement connexes de $\boldsymbol{V}(P) \cap \mathbb{R}^n$. Ceci constitue ce que nous appellerons par la suite le pré-traitement du problème.

On peut remarquer que les idéaux que l'on a construit sont tous engendrés par des polynômes de $\mathbb{Q}[X_1, ..., X_n]$ sauf éventuellement un qui est $P - \epsilon$. Ils peuvent donc être vus comme des idéaux de $\mathbb{Q}(\epsilon)[X_1, ..., X_n]$ ou encore comme extensions d'idéaux de $\mathbb{Q}[\epsilon][X_1, ..., X_n]$, ϵ pouvant être considéré comme une variable.

Une conséquence est que l'on peut, en pratique, calculer de plusieurs façons différentes la dimension de I :

- par le biais d'une base de Gröbner pour n'importe quel ordre en travaillant dans $\mathbb{Q}\langle\epsilon\rangle[X_1, ..., X_n]$;
- par le biais d'une base de Gröbner pour n'importe quel ordre en travaillant dans $\mathbb{Q}(\epsilon)[X_1, ..., X_n]$, ϵ étant alors considéré comme une variable;
- En remarquant qu'une base de Gröbner de $I \subset \mathbb{Q}[\epsilon, X_1, ..., X_n]$, ϵ étant alors considéré comme une variable, pour n'importe quel ordre sur $[\epsilon, X_1, ..., X_n]$ éliminant ϵ, est une base de Gröbner de $I \subset \mathbb{Q}(\epsilon)[X_1, ..., X_n]$ pour l'ordre induit sur $[X_1, ..., X_n]$;

L'algorithme suivant réalise le pré-traitement du problème évoque plus haut :

Algorithm HyperPreProcessing

- **Input** : $P \in \mathbb{Q}[X_1, ..., X_n]$, $I \to \dim(I)$ which computes the dimension of any ideal $I \subset \mathbb{Q}(\epsilon)[X_1, ..., X_n]$
- **Output** : a non positive dimensional system $\mathcal{S}_E \subset \mathbb{Q}(E)[X_1, ..., X_n]$ such that $\lim_0(\boldsymbol{V}(\mathcal{S}_\epsilon)) \cap \mathbb{R}^n$ is empty or intersects each semi-algebraically connected component of $\boldsymbol{V}(P) \cap \mathbb{R}^n$.
- if $\dim(\langle P, \frac{\partial P}{\partial X_i}, i = 1...n\rangle) > 0$ then $P := P - \epsilon$
- while $\dim(\langle P, P_{1,A}, ..., P_{r,A}\rangle) > 0$ in $\mathbb{Q}(\epsilon)[X_1, ..., X_n]$, choose $A \in \{0, ...d\}^n$
- return($\{P, P_{1,A}, ..., P_{r,A}\}$)

Une fois que l'on a déterminé le système zéro-dimensionnel $\mathcal{S}_\epsilon$ à résoudre, par exemple en appliquant l'algorithme HYPERPREPROCESSING, il « ne reste plus » qu'à calculer les limites quand $\epsilon \to 0$ de ses zéros bornés pour obtenir un point par composante semi-algébriquement connexe de $\boldsymbol{V}(P) \cap \mathbb{R}^n$. Comme dans le cas des systèmes à coefficients rationnels, l'option prise dans **FR**-[118] est de passer par le calcul d'une RUR de $\langle\mathcal{S}_\epsilon\rangle$.

On retrouve les trois problèmes déjà abordés que sont la prédiction d'une forme linéaire t_ϵ séparant $\boldsymbol{V}(\langle \mathcal{S}_\epsilon \rangle)$, le calcul d'un candidat RUR et la vérification a posteriori du choix de l'élément séparant.

En première approximation, on peut admettre que l'on sait calculer un candidat RUR et que l'on sait vérifier a posteriori si un élément est séparant ou non, par l'un des algorithmes proposés eu chapitre 3, dès lors que l'on connaît une description de $\frac{\mathbb{Q}\langle\epsilon\rangle[X_1,...,X_n]}{\langle \mathcal{S}_\epsilon \rangle}$ (par exemple connaissant une base de Gröbner de $\langle \mathcal{S}_\epsilon \rangle$).

En calculant un candidat RUR, on doit prendre quelques précautions pour faciliter l'étude des points de $\boldsymbol{V}(\mathcal{S}_\epsilon)$ dont la limite quand $\epsilon \to 0$ est bornée. En particulier, les multiplicités des limites de zéros de $\langle \mathcal{S}_\epsilon \rangle$ ne coïncideront pas, en général, avec celles des zéros de $\langle \mathcal{S}_0 \rangle$. Pour permettre l'analyse de l'évolution de ces multiplicités, on introduit 2 variantes de paramétrisations rationnelles pouvant se déduire d'une RUR :

Notation 4.5. *Soit* $\{f_{t,\epsilon}(T),\, g_{t,1,\epsilon}(T),\, g_{t,X_1,\epsilon}(T),\, ...,\, g_{t,X_n,\epsilon}(T)\} \subset \mathbb{Q}(\epsilon)[T]$ *une RUR de* $\langle \mathcal{S}_\epsilon \rangle$, $f_{t,\epsilon}$ *étant normalisé.*

On pose $g_{t,\epsilon} = \prod_{\alpha \in \boldsymbol{V}(\langle \mathcal{S}_\epsilon \rangle)} (T - t(\alpha))^{\mu(\alpha)-1} = \gcd(f_{t,\epsilon}, \frac{\partial f_{t,\epsilon}}{\partial T})$ *(normalisé) et, pour* $v = 1, X_1, ..., X_n$, $\tilde{g}_{t,v,\epsilon} = g_{t,\epsilon}\, g_{t,v,\epsilon}$. *et on notera* $A-\mathrm{RUR}$ *l'ensemble* $\{f_{t,\epsilon}(T),\, \tilde{g}_{t,1,\epsilon}(T),\, \tilde{g}_{t,X_1,\epsilon}(T),\, ...,\, \tilde{g}_{t,X_n,\epsilon}(T)\}$

On définit $\nu \in \mathbb{N}$, *le plus petit entier tel qu'il existe* $c(\epsilon) \in \mathbb{Q}[\epsilon]$ *tel que* $\epsilon^\nu c(\epsilon) f_{t,\epsilon}(T) \in \mathbb{Q}[\epsilon][T]$, *et on pose* $F_{t,\epsilon} = \epsilon^\nu c(\epsilon) f_{t,\epsilon}$, *ainsi que* $G_{t,v,\epsilon} = \epsilon^\nu c(\epsilon) \tilde{g}_{t,v,\epsilon}$ *pour* $v = 1, X_1, ..., X_n$. *On notera* $N-\mathrm{RUR}$ *l'ensemble* $\{G_{t,\epsilon}(T), G_{t,1,\epsilon}(T), G_{t,X_1,\epsilon}(T), ..., G_{t,X_n,\epsilon}(T)\}$.

Chacune de ces paramétrisation permet de retrouver la RUR initiale, mais elle ne peuvent être utilisées brutalement pour évaluer les racines. En particulier, tous les polynômes autres que $f_{t,\epsilon}(T)$ ou $F_{t,\epsilon}(T)$ s'annulent en les zéros multiples de $\boldsymbol{V}(\langle \mathcal{S}_\epsilon \rangle)$. On peut contourner ceci en calculant les multiplicités $\mu(\alpha)$ des $\alpha \in \boldsymbol{V}(\langle \mathcal{S}_\epsilon \rangle)$ et en remarquant que, pour tout $\alpha \in \boldsymbol{V}(\langle \mathcal{S}_\epsilon \rangle)$, $\frac{g_{t,v,\epsilon}^{\mu(\alpha)-1}(t(\alpha))}{g_{t,1,\epsilon}^{\mu(\alpha)-1}(t(\alpha))} = \frac{g_{t,v,\epsilon}(t(\alpha))}{g_{t,1,\epsilon}(t(\alpha))}, v = X_1, ..., X_n$.

Pour préparer le calcul des limites de zéros bornés, il faut s'assurer que si t est une forme séparante pour $\boldsymbol{V}(\langle \mathcal{S}_\epsilon \rangle)$, alors les images de 2 points infinitésimalement proches sont également infinitésimalement proches :

Définition 4.6. ***FR**-[118] On appelle élément bien séparant, toute forme linéaire* $t \in \mathbb{Q}[X_1, ..., X_n]$ *séparant* $\boldsymbol{V}(\langle \mathcal{S}_\epsilon \rangle)$ *et telle que :*

- $G_{t,v,\epsilon}(T) \in \mathbb{Q}[\epsilon][X_1, ..., X_n], v = 1, X_1, ..., X_n$;
- *si* $\alpha, \beta \in \boldsymbol{V}(\langle \mathcal{S}_\epsilon \rangle)$ *sont infinitésimalement proches, alors* $t(\alpha)$ *et* $t(\beta)$ *le sont également.*

Comme dans le cas de la recherche d'un élément séparant pour un système à coefficients rationnels, celle d'un élément bien séparant peut s'effectuer en parcourant une famille finie :

Lemme 4.7. ***FR**-[118] L'ensemble* $\{\sum_{i=0}^{n-1} j^{(i-1)}, j = 1...n\, (\#\boldsymbol{V}(\langle \mathcal{S}_\epsilon \rangle))\}^2$ *contient au moins un élément bien séparant pour* $\boldsymbol{V}(\langle \mathcal{S}_\epsilon \rangle)$.

En supposant que l'on connaisse une RUR de $\langle \mathcal{S}_\epsilon \rangle$, reste maintenant à calculer les limites quant $\epsilon \to 0$ des zéros de $\boldsymbol{V}(\langle \mathcal{S}_\epsilon \rangle)$ qu'elle représente et à discriminer ceux d'entre eux qui ont au moins une coordonnée non bornée :

Lemme 4.8. ***FR**-[118] Supposons que t soit un élément bien séparant, alors :*

- $f_{t,\epsilon}$ *admet des racines non bornées si et seulement si* $\deg(F_{t,0}(T)) < \deg(F_{t,\epsilon}(T))$;
- *Si* $\alpha \in \boldsymbol{V}(f_{t,\epsilon})$, *et que* α *est bornée, alors* $a = \lim_{\epsilon \to 0}(\alpha)$ *est une racine de* $F_{t,0}$.
- *Si* a *est une racine de multiplicité* μ *de* $F_{t,0}(T)$, *alors,* $\exists \alpha \in \boldsymbol{V}(f_{t,\epsilon})$, $\lim_{\epsilon \to 0}(\alpha) = a$ *et ,* $\forall \beta \in \boldsymbol{V}(f_{t,\epsilon})$ *de multiplicité* μ' *te telle que* $\lim_0(\beta) = a$, $\lim_{\epsilon \to 0}\left(\frac{g_{t,v,\epsilon}^{\mu'}(\beta)}{g_{t,v,\epsilon}^{\mu'}(\beta)}\right) = \frac{G_{t,v,0}^{\mu}(a)}{G_{t,v,0}^{\mu}(a)}, v = X_1, ..., X_n$.

Ainsi, si l'on connaît une RUR de $\langle \mathcal{S}_\epsilon \rangle$, le calcul de la N-RUR associée et la résolution de $F_{t,0}(T) = 0$ permettent d'obtenir toutes les racines bornées de $\mathcal{S}_\epsilon$ et d'en calculer les limites quant $\epsilon \to 0$, autrement dit de calculer un point au moins par composante semi-algébriquement connexe de $\boldsymbol{V}(P) \cap \mathbb{R}^n$.

Plusieurs optimisations sont proposées dans **FR**-[118] pour la vérification de l'élément séparant, la détection et la résolution du cas particulier ou $\langle \mathcal{S}_\epsilon \rangle$ est radical. La plus aboutie est certainement l'utilisation d'une arithmétique tronquée permettant de limiter le degré en ϵ des polynômes apparaissant dans les calculs, mais nous ne les détaillerons pas dans ce document, la portée pratique de la méthode générale étant loin d'être satisfaisante, en particulier si l'objectif est de traiter non plus des variétés définies par un unique polynôme mais des variétés définies par un système quelconque.

En effet, bien que l'on ait montré, dans **FR**-[118], l'apport de ce type de méthodes par rapport à la Décomposition Cylindrique Algébrique, les calculs engendrés restent suffisamment délicats pour que l'on se tourne vers une autre stratégie. En dehors des problèmes d'éléments séparants, si l'on ne tronque pas les degrés en ϵ, la RUR de $\boldsymbol{V}(\langle \mathcal{S}_\epsilon \rangle)$ est un objet très gros, difficile à calculer (voir chapitre 6 - problème d'interpolation de Brikhoff). Si on tronque les degrés en ϵ, on doit alors se passer de nombreuses simplifications comme la réduction des polynômes en cours de calcul par leur contenus : ce sont cette fois les scalaires dont la taille grandit énormément.

L'application à une variété non plus définie par un unique polynôme mais par un système quelconque accentue la difficulté puisque le fait de prendre la somme des carrés des équations augmente artificiellement le degré des polynômes à étudier mais surtout nous place quasi systématiquement dans le cas le plus général (nécessitant une déformation infinitésimale).

4.1.2 Le cas des variétés définies par un système d'équations.

Si l'on revient sur le problème de la section précédente, le constat est que la situation est bloquée, d'un point de vue calcul, dès lors que $\boldsymbol{V}(P)$ admet une infinité de points singuliers.

L'idée, développée et généralisée dans cette section, est d'utiliser le fait que $\boldsymbol{V}(\mathcal{J}) = \boldsymbol{V}(\langle P, \frac{\partial P}{\partial X_1}, ..., \frac{\partial P}{\partial X_n} \rangle)$ est une variété algébrique de dimension strictement inférieure à celle de $V(P)$ (si P est irréductible) et que $\boldsymbol{V}(\mathcal{C}(A)) = \{M \in \mathbb{C}^n,\, P(M) = 0 \wedge \mathrm{grad}_M(P) // \overrightarrow{\mathrm{AM}}\}$ est l'union de $\boldsymbol{V}(\langle P, \frac{\partial P}{\partial X_1}, ..., \frac{\partial P}{\partial X_n} \rangle)$ et d'un nombre fini de points. Le principe de base consiste alors à calculer $\boldsymbol{V}(\mathcal{C}(A)) \setminus \boldsymbol{V}(\mathcal{J})$ (en résolvant un système zéro-dimensionnel) puis à calculer un point par composante semi algébriquement connexe de la variété définie par les zéros de $\mathcal{J}$, qui est de dimension plus petite.

Une première remarque est que, même si l'entrée est une variété définie par un unique polynôme, on se retrouve dans le cas général dès lors que l'on doit étudier $\boldsymbol{V}(\mathcal{J})$. Dans [7], le premier travail a donc été de modéliser les points critiques de la fonction distance à un point dans de cas d'une variété définie par un nombre arbitraire d'équations.

Notation 4.9. *Soit* $\mathcal{S} = \{p_1, ..., p_r\} \subset \mathbb{Q}[X_1, ..., X_n]$ *tel que* $\dim(\boldsymbol{V}(\langle \mathcal{S} \rangle)) = d$ *et* $A \in \mathbb{Q}^n$. *On note :*

- $V = \boldsymbol{V}(\langle \mathcal{S} \rangle) \subset \mathbb{C}^n$
- $\boldsymbol{V}(\mathcal{C}(A)) = \{M \in \boldsymbol{V}(\langle \mathcal{S} \rangle), \mathrm{rank}(\mathrm{grad}_M(p_1), ... \mathrm{grad}_M(p_r), \overrightarrow{\mathrm{AM}}) \leq n - d;$
- $\boldsymbol{V}(\mathcal{J}) = \{M \in \boldsymbol{V}(\langle \mathcal{S} \rangle), \mathrm{rank}(\mathrm{grad}_M(p_1), ... \mathrm{grad}_M(p_r)) < n - d.$

Les ensembles $\boldsymbol{V}(\mathcal{C}(A))$ et $\boldsymbol{V}(\mathcal{J})$ se construisent simplement à partir de mineurs de la matrice jacobienne du système, éventuellement augmentée d'une colonne contenant les coordonnées de $\overrightarrow{\mathrm{AM}}$:

Notation 4.10. *Pour* $B \in \mathbb{C}^n$ *et* $\{Q_1, ..., Q_s\} \subset \mathbb{C}[X_1, ..., X_n]$, *on note* $\mathcal{M}_B(Q) = \left(\left(\frac{\partial Q_j}{\partial X_i} \right)_{i=1...s}^{j=1...n} | \, \overrightarrow{\mathrm{BM}} \right)$ *et on définit, pour tout entier* k *:*

- $\Delta_{B,k}(Q)$, *l'ensemble des mineurs d'ordre* $n - k + 1$ *de* $\mathcal{M}_B(Q)$ *contenant des coordonnées de* $\overrightarrow{\mathrm{BM}}$;

- $\Gamma_{B,k}(Q)$, *l'ensemble des mineurs d'ordre* $n-k$ *de* $\mathcal{M}_B(Q)$ *ne contenant pas de coordonnées de* $\overrightarrow{\mathrm{BM}}$.

Avec ces notations, on peut par exemple définir $\mathcal{C}(A) = \mathcal{S} \cup \Delta_{B,d}$ et $\mathcal{J} = \mathcal{S} \cup \Gamma_{B,d}$.

L'ensemble $\boldsymbol{V}(\mathcal{C}(A))$ contient clairement les points critiques de la fonction « distance au point A ». En plus du problème de généricité de A, qui ne doit être ni sur la variété, ni équi-distant d'une infinité de points de V, un écueil supplémentaire est que $\boldsymbol{V}(\mathcal{J})$ peut ne pas contenir tous les points singuliers de V (par exemple ceux contenues dans des composantes irréductibles de petite dimension). Même en supposant V équi-dimensionnelle, $\boldsymbol{V}(\mathcal{J})$ peut contenir beaucoup d'autres points comme par exemple certains points de certaines composantes immergées (ça n'est pas parce que V est une variété équi-dimensionnelle que $\langle p_1, \ldots, p_s \rangle$ est un idéal équi-dimensionnel) au point de pouvoir être de la même dimension que V (par exemple si tous les p_i sont des carrés). Modulo quelques conditions supplémentaires, les résultats obtenus dans le cas d'une hyper-surface se généralisent bien :

Théorème 4.11. *[7] Si on suppose que* $\langle p_1, \ldots, p_r \rangle$ *est équi-dimensionnel et radical, alors, il existe un entier* D *et un point* $A \in \{1, \ldots, D\}^n$ *tels que :*

- $\boldsymbol{V}(\mathcal{C}(A))$ *intersecte chaque composante semi-algébriquement connexe de* $V \cap \mathbb{R}^n$*;*
- $\boldsymbol{V}(\mathcal{C}(A)) = \boldsymbol{V}(\mathcal{J}) \cup V_{0,A}$ *avec* $\#V_{0,A} < \infty$*;*
- $\dim(\boldsymbol{V}(\mathcal{J})) < \dim(V)$*;*

La condition de généricité pour le point A sous-entendue par ce théorème est similaire à celle de la section précédente : il suffit de s'assurer que $\dim(\boldsymbol{V}(\mathcal{C}(A)) < \dim(V)$.

Si on suppose que l'on dispose d'un algorithme permettant de décomposer n'importe quel idéal S de telle sorte que $\boldsymbol{V}(\langle S \rangle) = \cup_{i=1}^{k} \boldsymbol{V}(\langle S_i \rangle)$, les S_i étant des systèmes tels que $\langle S_i \rangle$ soient équi-dimensionnels et radicaux (en utilisant par exemple les algorithmes de décomposition proposés dans [16]), on a alors, en principe, un algorithme simple pour le calcul d'un point par composante semi-algébriquement connexe de $V \cap \mathbb{R}^n$:

Algorithm Critical

- **Input** : $I \subset \mathbb{Q}[X_1, \ldots, X_n]$
- **Output** : a set of zero-dimensional systems I_i such that $\cup_{i=1}^{k} \boldsymbol{V}(I_i) \cap \mathbb{R}^n$ meets each semi-algebraically connected component of $\boldsymbol{V}(I)$
- Result $:= \emptyset$;
- Todo $=$ Decompose(I);
- while(Todo $\neq \emptyset$)
  - $I =$ First(Todo); Todo $:=$ Todo $\setminus I$;
  - $d = \dim(I)$;
  - if $\dim(I) = 0$ Result $:=$ Result $\cup \{I\}$
  - else
    - (*) Choose $A \in \{1, \ldots, D\}^n$ (start with $D = 1$ and increase D when all the choices failed)
    - $I' = \langle \Delta_{A,d} \rangle + I$;
    - if $\dim(I') = d$ goto (*)
    - else Result $:=$ Result $\cup$ Critical(I')
- Return(Result);

Bien que cela n'apparaisse pas directement dans la description de l'algorithme, deux calculs peuvent être, en pratique, extrêmement coûteux :

- la décomposition de I;
- le calcul de $\Delta_{A,d}$

La décomposition de I est de toute évidence une étape critique de l'algorithme de part son coût mais également parce qu'elle conditionne le calcul de $\Delta_{A,d}$. Si on utilise par exemple les algorithmes proposés dans [16], les composantes $I_i, i=1...k$ seront données sous la forme de bases de Gröbner : le nombre de générateurs de chaque I_i n'a donc aucune raison d'être de de l'ordre de $n-\dim(I_i)$, entraînant un facteur combinatoire éventuellement important dans le nombre d'éléments de $\Delta_{A,d}$, puisque, si s est le nombre de générateurs de I_i, il faudra calculer $\binom{s}{n-d}\binom{n}{n-d+1}$ mineurs.

Dans [7], les $I_i, i=1...k$ sont calculés par une décomposition en ensembles triangulaires réguliers et séparables [6].

Un ensemble triangulaire de $\mathbb{Q}[X_1, ..., X_n]$ de dimension d est un ensemble de polynômes de la forme $\mathcal{T}=\{f_{d+1}(X_1, ..., X_{d+1}), ..., f_n(X_1, ..., X_n)\}$. Il est dit *régulier* si le coefficient de tête de chaque f_i en la variable X_i ne s'annule pas sur l'une des composantes irréductibles de $\boldsymbol{V}(f_1, ..., f_{i-1})$. Il est dit *séparable* lorsque chaque $\frac{\partial f_i}{\partial X_i}$ ne s'annule pas sur l'une des composantes irréductibles de $\boldsymbol{V}(f_1, ..., f_{i-1})$.

Sans rentrer dans les détails, toute variété $\boldsymbol{V}(I)$ de dimension d peut être vue comme une union $\cup_{i=1}^{k}\boldsymbol{V}(\mathrm{sat}(\mathcal{T}_i))$, les $\mathcal{T}_i$ étant des ensembles triangulaires et $\mathrm{sat}(\mathcal{T}_i)=\langle\mathcal{T}_i: h^\infty\rangle$, h étant le produit des coefficients de tête des f_i en X_i. Lorsque les $\mathcal{T}_i$ sont réguliers et séparables, alors les idéaux $\mathrm{sat}(\mathcal{T}_i)$ sont équi-dimensionnels et radicaux [6]. Une telle décomposition peut se calculer directement ou en utilisant des bases de Gröbner de manière efficace (en tout cas comparativement aux autres décompositions plus fines).

Théoriquement , la fonction DECOMPOSE doit calculer les idéaux $\mathrm{sat}(\mathcal{T}_i)$, mais l'optimisation proposée dans [7] utilise plutôt directement les $\mathcal{T}_i$ pour le calcul de $\mathcal{C}(A)$:

Notation 4.12. *Pour $A=(a_1, ..., a_n)\in\mathbb{C}^n$, $Q\subset\mathbb{Q}[X_1, ..., X_n]$ et $\mathcal{T}=\{t_{d+1}, ..., t_n\}\subset\mathbb{C}[X_1, ..., X_n]$, un ensemble triangulaire tel que $\langle Q\rangle=\mathrm{sat}(\mathcal{T})$, on note*

$$\mathcal{M}_B^{(k)}(Q)=\left(\begin{array}{c|c}\left(\frac{\partial t_j}{\partial X_k}\right)_{j=d+1...n} & X_k-a_k \\ \hline \left(\frac{\partial t_j}{\partial X_k}\right)^{i=d+1...n}_{j=d+1...n} & \begin{array}{c}X_{d+1}-a_{d+1}\\ \vdots \\ X_n-a_n\end{array}\end{array}\right)$$

et on définit, pour tout entier k, $\Delta'_{A,d}=\{\det(\mathcal{M}_B^{(k)}(Q), k=1...d\}$

L'idée utilisée dans [7] est de remplacer $\Delta_{A,d}$ par $\Delta'_{A,d}$ dans l'algorithme CRITICAL de sorte à accélérer les calculs : il n'y a que d mineurs à considérer et leur calcul est grandement simplifié par la structure triangulaire des systèmes utilisés. Le résultat qui suit montre que l'opération est possible, même si cette modification entraîne le calcul d'un peu plus de points :

Proposition 4.13. *[7] Soient $A=(a_1, ..., a_n)\in\mathbb{C}^n$, $Q\subset\mathbb{Q}[X_1, ..., X_n]$ et $\mathcal{T}=\{t_{d+1}, ..., t_n\}\subset\mathbb{C}[X_1, ..., X_n]$, un ensemble triangulaire tel que $\langle Q\rangle=\mathrm{sat}(\mathcal{T})$. Si on note $\mathcal{D}(A)$ l'idéal $\langle Q\rangle+\Delta'_{A,d}$, et $\mathrm{Sep}(\mathcal{T})=\{\frac{\partial t_i}{\partial X_i}, i=d+1...n\}$, alors :*

- $\boldsymbol{V}(\mathcal{C}(A))\subset\boldsymbol{V}(\mathcal{D}(A))$;
- $\boldsymbol{V}(\mathcal{D}(A))=V'_{0,A}\cup\boldsymbol{V}(\langle\mathrm{Sep}(\mathcal{T})\rangle)$ *avec $\#V'_{0,A}<\infty$ si A n'est pas équidistant d'une infinité de points de $\boldsymbol{V}(\langle Q\rangle)$;*
- $\dim(\langle Q\rangle+\langle\mathrm{Sep}(\mathcal{T})\rangle)<\dim(Q)$;

En particulier, $\boldsymbol{V}(\mathcal{D}(A))\cap\mathbb{R}^n$ intersecte chaque composante semi-algébriquement connexe de $\boldsymbol{V}(\langle Q\rangle)$.

Dans [7], diverses optimisations sont proposées dans le cas d'utilisation d'ensembles triangulaires. Par exemple, il est courant dans les calculs d'avoir à faire à des ensembles du type $\mathcal{T} = \{t_{d+1}, ..., t_n\}$ ou les $t_i, i = d+2...n$ sont linéaires en les X_i : l'astuce consiste on alors à « oublier » ces polynômes partout ou ils ne sont pas identiquement nuls.

Dans ce qui suit, on évalue sur quelques exemples les performances de deux variantes de l'algorithme Critical :

- Critical : une implantation de l'algorithme présenté dans cette section utilisant les ensembles triangulaires pour la fonction Decompose et basée sur la proposition 4.13.
- IsEmpty : une variante de l'algorithme Critical se contentant de décider si une variété algébrique admet des points réels ou non. Par exemple, le calcul s'arrête dès qu'un point réel est trouvé.

Ces variantes utilisent une implantation de la décomposition en ensembles triangulaires réguliers et séparables basée sur des calculs de bases de Gröbner ainsi qu'une implantation de l'algorithme ZeroDimSolve du chapitre 3 pour la résolution des systèmes zéro-dimensionnels générés.

4.1.3 Expériences et conclusions

Le premier tableau est un rappel des expériences menées sur le problème d'interpolation de Birkhoff (voir Chapitre 6). L'algorithme CriticalHyp est une instance probabiliste de **FR**-[119] ou le calcul de RUR est effectué par diverses interpolations spécialement optimisées et récupérant d'ailleurs certains principes utilisés dans [87]. Dans les 2 cas, on ne mesure que le temps de calcul de la génération des systèmes zéro-dimensionnels en prenant pour unité le temps de calcul le temps d'exécution de l'algorithme Critical. La première colonne (Bounded) donne le nombre de points bornés dans CriticalHyp, la seconde colonne donne le nombre total de points (comptés avec multiplicités) dans CriticalHyp, la troisième colonne donne le nombre de points calculés par l'algorithme Critical et la dernière colonne donne le rapport de temps de calcul CriticalHyp/Critical

Hypersurface	Bounded	Degree (t)	Degrees(Critical)	CriticalHyp
Birk.3-1	12	16	12	70
Birk.3-2	7	16	7	40
Birk.3-3	15	34	15	86
Birk.3-4	16	36	16	255
Birk.3-5	31	40	31	252
Birk.3-6	37	52	37	173
Birk.3-7	38	52	38	159
Birk.3-8	45	130	45	552
Birk.3-9	47	132	47	373
Birk.3-10	48	136	48	2343
Birk.3-11	50	138	50	2988
Birk.3-12	50	138	50	1749
Birk.3-13	32	252	32	∞
Birk.3-14	60	264	60	∞
Birk.3-15	60	272	60	∞

Les scores sont sans appel et la conclusion sans ambiguité, d'autant qu'il s'agit d'hyper-surfaces, l'écart se creuse encore plus si on considère des systèmes quelconques puisqu'il faudra prendre la somme des carrés des équations, et donc élever artificiellement le degré, dans le cas de l'algorithme CriticalHyp.

Pour mesurer l'apport des variantes de l'algorithme CRITICAL en général, on utilise comme référentiel l'implantation a priori la plus aboutie de l'algorithme de Décomposition Cylindrique Algébrique : QEPCAD (bibliothèque SACLIB). L'unité de temps est le temps de calcul de l'algorithme ISEMPTY. Pour QEPCAD, la mention failed(n) indique que le calcul s'est arrêté après la construction de n cellules. Pour l'algorithme CRITICAL, la mention $t/(n)$ indique que t est le temps de calcul et n le nombre de points complexes comptés avec multiplicités des systèmes zéro-dimensionnels codant le résultat. L'unité de temps utilisée est le temps de calcul de IsEmpty. Les systèmes utilisés font partie des rares tests disponibles pour les systèmes de dimension positive, les 3 derniers exemples étant ceux provenant de l'application sur les bancs de filtres (voir Chapitre 6).

System	**Dim.**	**Deg.**	**Nb Vars**	**CRITICAL**	**QEPCAD**
Werneer	1	26	5	332 (84)	4300
Wang	1	114	13	10.5 (132)	∞
Euler	3	2	10	1 (10)	failed(872043)
Neural	1	24	4	2.31 (133)	2.0
Butcher	3	3	8	1 (15)	∞
Buchberger	4	6	8	1 (32)	failed(991324)
DiscPb	2	3	4	10 (28)	∞
Donati	1	10	4	250 (61)	0.6
Hairer2	2	25	13	∞	failed(872043)
Prodecco	2	2	5	∞	∞
F633	2	32	10	∞	∞
F744	1	40	12	∞	∞
F855	1	52	14	∞	∞

L'apport des méthodes de points critiques est indéniable. Au delà des temps de calcul, il faut surtout remarquer la taille de la sortie qui reste modérée. Par exemple, sur le système EULER, la CAD s'arrête en estimant à plusieurs centaines de milliers le nombre de cellules à traiter , et donc le nombre de points en sortie alors que l'algorithme CRITICAL fournit au plus 10 points.

L'autre remarque est que le nombre de variables n'influe pas sur la taille de la sortie dans le cas de l'algorithme CRITICAL, ce qui n'est mécaniquement pas le cas de la CAD.

Ce qui n'apparaît pas clairement dans le tableau, c'est que l'algorithme IsEmpty parvient à décider si tous ces systèmes admettent ou non des zéros réels.

Aucun des algorithmes proposés dans ce chapitre n'appartient à la bonne classe de complexité « dans le pire cas », à cause, principalement, des calculs de décompositions d'idéaux (les calculs de bases de Gröbner sont, dans ce cas, doublement exponentiels dans le pire cas). Par contre, il a été montré ultérieurement en utilisant, en particulier, les résultats de [127], que la taille de la sortie (nombre de points calculés) est correcte (polynomiale en d^n, d étant les degrés des équations), ce qui rassure sur le bien fondé de l'approche.

Chapitre 5
Systèmes dépendant de paramètres

5.1 Introduction

Bien que les résultats présentés dans ce chapitre soient les plus récents de ce document, les systèmes paramétrés constituent la classe la plus importante des problèmes soumis, la présence de paramètres dans un système ou une équation incitant naturellement à se tourner vers des outils de calcul formel. Afin d'éviter de nombreuses paraphrases, les notations suivantes seront systématiquement utilisées par la suite :

Notation 5.1. *Selon que l'on se place dans* $\mathbb{R}$ *ou dans* $\mathbb{C}$*, on étudie les ensembles*

$$\mathcal{S} = \{x \in \mathbb{R}^n \quad , \quad p_1(x) = 0, ..., p_s(x) = 0, f_1(x) > 0, ... f_l(x) > 0\}$$

ou

$$\mathcal{C} = \{x \in \mathbb{C}^n \quad , \quad p_1(x) = 0, ..., p_s(x) = 0, f_1(x) \neq 0, ... f_l(x) \neq 0\}$$

les p_i, f_j *étant des polynômes à coefficients rationnels.*

- $[U, X] = [U_1, ...U_d, X_{d+1}, ...X_n]$ *est l'ensemble des indéterminées,* $U = [U_1, ...U_d]$ *celui des paramètres et* $X = [X_{d+1}, ...X_n]$ *celui des inconnues;*
- *On note* $\mathcal{E} = \{p_1, ...p_s\}$ *et* $\mathcal{F} = \{f_1, ... f_l\}$ *les deux ensembles de polynômes utilisés.*
- *La substitution d'un sous-ensemble d'indéterminées* Y *par un vecteur de nombres complexes* y *dans les polynômes définissant un ensemble* $\mathcal{D}$ *sera notée* $\mathcal{D}_{|Y=y}$.
- *La fonction de projection sur l'espace des paramètres sera notée* Π_U: $\begin{array}{ccc} \mathbb{C}^n & \longrightarrow & \mathbb{C}^d \\ (u_1, ...u_d, x_{d+1}, ..., x_d) & \mapsto & (u_1, ..., u_d) \end{array}$

Dans les applications, les systèmes fréquemment posés sont « génériquement zéro-dimensionnels », c'est à dire que pour presque toutes les valeurs de paramètres, le système d'équations sous-jacent $\mathcal{E}$, une fois instancié, admet un nombre fini de solutions complexes; autrement dit, pour presque tout $u \in \mathbb{C}^d$, $\{p_1(u, X) = 0, ..., p_s(u, X) = 0\}$ admet un nombre fini de solutions dans $\mathbb{C}^{n-d}$. En effet, il est habituel, dans les domaines applicatifs, que les systèmes soient posés de telle façon qu'une méthode numérique simple puisse résoudre la plupart des spécialisations. Si on prend le cas le plus courant, tout est mis en oeuvre (choix des équations en particulier) pour que la méthode de Newton (par exemple) puisse s'appliquer sur presque toutes les instances. En général, ceci se traduit, avec nos notations, par le fait que $\#\mathcal{E} = n - d$ et que que l'idéal $\langle \mathcal{E}_{U=u} \rangle$ est zéro-dimensionnel et radical pour presque toutes les valeurs de $u \in \mathbb{C}^d$.

En première approximation, il est donc raisonnable de se restreindre aux idéaux $\mathcal{I}$ zéro-dimensionnels dans $\mathbb{Q}(U)[X]$. Mathématiquement, on peut alors récupérer la plupart des résultats du chapitre 3 et estimer résolu le problème puisque l'on peut alors calculer une Représentation Univariée Rationnelle de $\mathcal{I}$ vu comme idéal de $\mathbb{Q}(U)[X]$, ou toute autre forme de paramétrisation rationnelle comme par exemple celle proposée dans [128]. Il est facile de démontrer que la spécialisation telle RUR « générique » est une RUR pour presque toutes les spécialisations du système étudier. Plus précisément, si $f_t(U, T)$, $g_{t,1}(U, T)$, $g_{t,X_1}(U, T)$, ..., $g_{t,X_n}(U, T)$ est une RUR « générique » et que $u = u_1, ..., u_d$ est un d-uplet de complexes tels que le nombre de solutions distinctes de $f_t(u, T)$ est maximal, alors $f_t(u, T), g_{t,1}(u, T), g_{t,X_1}(u, T), ..., g_{t,X_n}(u, T)$ est une RUR de $\langle \mathcal{E}_{U=u} \rangle \subset \mathbb{Q}[X]$.

Ce type d'objet s'adapte donc, en théorie, très bien au cas des systèmes paramétrés, mais les objections pour ne pas l'utiliser sont multiples :

- en pratique, il est très gros, puisqu'à la croissance des coefficients induite par le calcul que l'on peut observer dans le cas zéro-dimensionnel non paramétrique, il faudra ajouter l'élévation des degrés des coefficients, qui sont des polynômes de $\mathbb{Q}[U]$ (ou des fractions rationnelles selon la stratégie de calcul); Il faudra donc s'attendre à ne pourvoir traiter que des classes d'exemples admettant génériquement un faible nombre de solutions complexes comptées avec multiplicités (dimension de $\frac{\mathbb{Q}(U)[X_1,...,X_n]}{\mathcal{I}}$ comme $\mathbb{Q}(U)$- espace vectoriel.

- Il est d'autant plus gros que f_t accumule l'ensemble de l'information sur les solutions (par exemple celle sur les points singuliers) mais également beaucoup d'informations parasites :

  - $f_t(t)$ étant, par construction, la clôture de Zariski d'une projection, certains points singuliers de f_t ne sont pas de liens avec des points singuliers de $\boldsymbol{V}(\langle p_1,...,p_s\rangle)$.

  - lorsque le nombre de solutions distinctes def_t chute pour une valeur des paramètres, ce peut être dû à une racine multiple de f_t, mais également à une valeur des paramètres pour laquelle la forme linéaire t n'est pas séparante, ce qui est lié à l'algorithme et non à l'objet étudié.

  - ces valeurs singulières « parasites » sont difficiles, voir impossibles, à discriminer, l'objet sera artificiellement compliqué.

- le traitement de cas généraux, c'est à dire ne supposant pas que $\mathcal{I}$ est zéro-dimensionnel dans $\mathbb{Q}(U)[X]$, semble difficile.

Même si il est, en apparence, tentant de calculer une « formule » générique (par exemple une RUR), fournir ce type d'objet n'est a priori pas la bonne façon de procéder, que ce soit d'un point de vue pratique ou d'un point de vue théorique. Une remarque plus pragmatique est que, de toutes façons, une telle formule n'est qu'un objet intermédiaire : elle a toujours pour vocation d'être étudiée soit pour obtenir une information partielle (existence ou nombre de solutions en fonction des valeurs de paramètres) pour garantir certaines propriétés (résistance des solutions à une déformation, preuve ou génération d'un processus numérique, etc.).

Si l'on replace la résolution des systèmes paramétrés dans un contexte global, c'est à dire tenant compte de l'utilisation qui sera faite de la « solution » fournie, on s'aperçoit alors que la question centrale est de caractériser les ouverts $\mathcal{U}$ de l'espace des paramètres au dessus desquels l'ensemble étudié est « régulier » : nombre constant de solutions, stabilité des solutions, applicabilité du théorème des fonctions implicites, etc. Dans le cas d'ensembles constructibles ou semi-algébriques basiques, une bonne notion est d'assurer que $(\Pi_U^{-1}(\mathcal{U}) \cap \mathcal{C}, \Pi_U)$ est un revêtement analytique de $\mathcal{U}$ **FR**-[86]. Si on suppose que u est un point d'un tel ouvert, la propriété de revêtement assure alors que le nombre de solutions du système est constant au voisinage de u et que toutes les feuilles du revêtement sont localement difféomorphes à $\mathcal{U}$. A titre d'exemple, si on étudiait les valeurs des paramètres pour lesquels le polynôme f_t d'une RUR « générique » admet un nombre maximal de solutions, on caractériserait de tels ouverts.

Quelle que soit la question posée, on cherche donc un ensemble fini d'ouverts $\mathcal{U}_i, i=1...l$ vérifiant cette propriété de revêtement, et dont l'union est, de préférence, dense dans $\Pi_U(\mathcal{C})$ ou $\Pi_U(\mathcal{S})$. Si l'on désire alors, par exemple, calculer le nombre (constant et fini) de solutions en tout point d'un $\mathcal{U}_i$, il suffit de prendre un seul point de cet ouvert, de substituer ses coordonnées aux paramètres U dans $\mathcal{S}$ et de résoudre ce système. Par construction, le nombre de zéros complexes communs aux équations ainsi obtenues est fini, et on peut donc, par exemple, utiliser l'algorithme du chapitre 3 pour le résoudre (et en particulier compter le nombre de solutions).

Il existe quelques solutions algorithmiques permettant de caractériser de tels ouverts, on peut songer aux paramétrisations rationnelles (RUR, résolution géométrique [128]), aux décomposition en ensembles triangulaires réguliers et séparables (voir [6] pour une vue homogènes et [142] pour une introduction générale) et bien sûr à la décomposition cylindrique algébrique [30]. Dans les cas on projettera les « défauts » de ces représentations sur l'espace des paramètres, c'est à dire un ensemble W de valeurs de paramètres contenant tous ceux pour lesquels il n'existe pas de voisinage ayant la « propriété de revêtement » décrite plus haut : dans le cas d'une paramétrisation rationnelle ce seront les paramètres pour lesquels le nombre de solutions distinctes du premier polynôme (noté f_t plus haut) chute, dans le cas d'un ensemble triangulaire, calculé pour n'importe quel ordre lexicographique éliminant en priorité les indéterminées, on rajoutera les valeurs des paramètres telles que le nombre de solutions complexes des autres polynômes varie (voir Chapitre 4 pour quelques informations supplémentaires). Dans le cas de l'utilisation d'une CAD dont les premières étapes de la phase de projection éliminent les indéterminées X_{d+1}, ..., X_n, on peut considérer l'ensemble des zéros des polynômes obtenus après $n-d$ étapes de projection. Sur ces exemple, les ouverts $\mathcal{U}$ recherchés seront les ouverts de $\Pi_U(\mathcal{C})$ qui ne rencontrent pas W.

On peut remarquer que toutes les solutions évoquées ci-dessus calculent une variété W qui dépend de l'algorithme utilisé (choix de formes séparantes pour les paramétrisations, ordre sur les variables pour les autres) : elle n'a que peu de chances d'être optimale en général.

5.2 Les variétés discriminantes dans le cas complexe

L'idée sous-jacente aux variétés discriminantes est de caractériser l'ensemble des points u de l'espace des paramètres tels qu'il existe un voisinage $\mathcal{U} \ni u$ tel que $(\Pi_U^{-1}(\mathcal{U}) \cap \mathcal{C}, \Pi_U)$ soit un revêtement analytique de $\mathcal{U}$.

Comme nous le justifierons plus formellement par la suite, on peut considérer la clôture algébrique $\bar{\mathcal{C}}$ de $\mathcal{C}$ au lieu de $\mathcal{C}$ lui-même, les points rajoutés se projetant sur des paramètres n'ayant pas la propriété recherchée. Ceci nous permettra, d'une part de ne manipuler que des ensembles algébriques, rendant certaines définitions et résultats plus faciles à exprimer, d'autre part de faciliter le lien avec les aspects calculatoires. Plus généralement, nous utiliserons la notation :

Notation 5.2. *Pour tout ensemble constructible $\mathcal{D} \subset \mathbb{C}^k$, on note $\overline{\mathcal{D}} \subset \mathbb{C}^k$ sa clôture algébrique.*

La démarche proposée dans **FR**-[86] consiste d'abord à caractériser les valeurs des paramètres tels qu'il n'existe aucun voisinage $\mathcal{U}$ tel que $(\Pi_U^{-1}(\mathcal{U}) \cap \mathcal{C}, \Pi_U)$ en soit un revêtement analytique. Certains peuvent être identifiés et regroupés facilement :

Notation 5.3. *On suppose que $\mathcal{C}$ est un ensemble constructible basique, de clôture algébrique $\bar{\mathcal{C}}$ et de dimension δ. Si δ' est la dimension de $\overline{\Pi_U(\mathcal{C})}$, on note :*

- *O_{sd}, la projection des composantes irréductibles de $\bar{\mathcal{C}}$ de dimension $< \delta$;*
- *O_c, les valeurs critiques « au sens large » de Π_U en restriction à $\bar{\mathcal{C}}$, c'est à dire la projection des points singuliers de $\bar{\mathcal{C}}$ et les valeurs critiques de Π_U (en restriction au lieu régulier de $\bar{\mathcal{C}}$).*
- *O_∞, les points $u \in \overline{\Pi_U(\mathcal{C})}$ tels que pour tout voisinage compact $\mathcal{U} \ni u$, $\Pi_U^{-1}(\mathcal{U}) \bigcap \bar{\mathcal{C}}$ n'est pas compact.*
- *$O_{\mathcal{F}}$ (resp. O_{F_i}) la projection de l'intersection de $\bar{\mathcal{C}}$ avec $\{x \in \mathbb{C}^n, 0 = \prod_{i=1}^s f_i(x)\}$ (resp. $\{x \in \mathbb{C}^n, 0 = f_i\}$);*
- *O_{sing} le lieu singulier de $\overline{\Pi_U(\mathcal{C})}$.*

On peut se convaincre facilement que les points ensembles O_{sd}, O_c, O_∞, $O_{\mathcal{F}}$ et O_{sing} ne vérifient pas la « propriété de recouvrement » recherchée, et il est de plus montré dans **FR**-[86] que tout point de l'espace des paramètres non contenu dans leur union vérifie cette propriété. Un point important pour l'aspect effectif est que cette union est un ensemble algébrique, ce qui n'est pas forcément le cas des ensembles pris séparément (hormis O_∞ et O_{sing}). Il faut noter que l'ensemble O_{sing} est vide dès lors que $\Pi_U(\mathcal{C})$ est vide ou dense dans $\mathbb{C}^d$, ce qui est le cas par exemple des systèmes « génériquement zéro-dimensionnels » évoqués en introduction de ce chapitre. On résume ceci par la proposition suivante :

Proposition 5.4. ***FR**-[86] L'ensemble $W_D = O_{\text{sd}} \cup O_c \cup O_\infty \cup O_{\mathcal{F}} \cup O_{\text{sing}}$ est un ensemble algébrique de $\mathbb{C}^d$ (contenu dans $\overline{\Pi_U(\mathcal{C})}$). Étant donné un ouvert $\mathcal{U} \subset \Pi_{\mathcal{U}}(\mathcal{C})$, $(\Pi_U^{-1}(\mathcal{U}) \cap \mathcal{C}, \Pi_U)$ est un recouvrement analytique de $\mathcal{U}$ si et seulement si $\mathcal{U} \cap W_D = \emptyset$.*

Sans rentrer dans les détails de la démonstration de ces résultats (donnés dans **FR**-[86]), on peut toutefois mentionner les deux principaux artifices utilisés.

Le premier est essentiellement lié au fait que la topologie de Zariski et la topologie usuelle coïncident pour $\mathbb{C}^n$:

Lemme 5.5. *L'ensemble O_∞ est un fermé de Zariski (définissable comme ensemble de zéros d'un idéal de $\mathbb{Q}[U]$). Plus précisément $W_\infty := \pi(\bar{\mathcal{C}}^{\,\mathbb{P}} \bigcap \mathcal{H}_\infty)$, avec :*

- *$\mathbb{P}^{n-d}$ l'espace projectif associé à $\mathbb{C}^{n-d}$;*
- *$\bar{\mathcal{C}}^{\,\mathbb{P}}$ la clôture projective de $\mathcal{C}$ dans $\mathbb{C}^d \times \mathbb{P}^{n-d}$;*
- *$\mathcal{H}_\infty$ l'hyperplan à l'infini de $\mathbb{C}^d \times \mathbb{P}^{n-d}$, i.e. $\mathcal{H}_\infty = \left(\mathbb{C}^d \times \mathbb{P}^{n-d}\right) \setminus \left(\mathbb{C}^d \times \mathbb{C}^{n-d}\right)$;*
- *π la projection canonique $\mathbb{C}^d \times \mathbb{P}^{n-d} \longrightarrow \mathbb{C}^d$.*

Le deuxième montre un système de vases communicants entre les composantes, permettant un certain nombre d'écarts dans les formulations ainsi que dans les calculs effectifs :

Lemme 5.6. *Avec les notations de la proposition 5.4*

- $O_\infty = \overline{O_\infty}$
- $\overline{\Pi_U(\mathcal{C})} \setminus \Pi_U(\mathcal{C}) \subset O_\infty$
- *Pour tout ensemble constructible $O \subset \Pi_U(\mathcal{C})$, $\overline{O} \setminus O \subset O_\infty$*

La proposition 5.4 donne donc une caractérisation algébrique des ensembles de points que nous recherchons comme complémentaire d'une variété algébrique. Cette variété est intrinsèque (définie en dehors de tout algorithme) et optimale (il n'en existe pas de plus petite) :

Définition 5.7. ***FR**-[86] On appelle variété discriminante minimale de $\mathcal{C}$ pour Π_U l'ensemble $O_{\text{sd}} \cup O_c \cup O_\infty \cup O_{\mathcal{F}} \cup O_{\text{sing}}$.*

Le qualificatif « minimale » rappelle le caractère optimal de cette variété mais on peut toutefois étendre la définition pour relâcher l'aspect minimal :

Définition 5.8. *Une variété algébrique $W \subset \mathbb{C}^d$ est une variété discriminante de $\mathcal{C}$ pour Π_U si et seulement si :*

- $W \subset \overline{\Pi_U(\mathcal{C})}$
- $W = \overline{\Pi_U(\mathcal{C})}$ *si et seulement si $\Pi_U^{-1}(u) \cap \mathcal{C}$ est infini ou vide pour presque tout $u \in \Pi_U(\mathcal{C})$;*
- *Les composantes connexes $\mathcal{U}_1, \ldots \mathcal{U}_k$ de $\overline{\Pi_U(\mathcal{C})} \setminus W$ sont des sous-variétés analytiques de dimension δ (cette condition est automatiquement vérifiée lorsque $\overline{\Pi_U(\mathcal{C})} = \mathbb{C}^d$)*
- *Pur $i = 1 \ldots k$, $(\Pi_U^{-1}(\mathcal{U}_i) \bigcap \mathcal{C}, \Pi_U)$ est un revêtement analytique de $\mathcal{U}_i$.*

Pour revenir à l'exploitation d'une RUR « générique » abordée en introduction, l'union des zéros du terme de tête de f_t et des zéros du discriminant de f_t définit une variété discriminante. Elle ne peut cependant pas être minimale en général puisqu'elle contiendra « en plus » les valeurs des paramètres pour lesquels l'élément séparant « générique » ne sépare pas les zéros du système spécialisé en ces valeurs.

5.3 Les variétés discriminantes dans le cas réel

Dans le cas réel, on peut définir de manière équivalente les ensembles O_{sd}, O_c, O_∞, $O_{\mathcal{F}}$ et O_{sing}, mais par contre, leur union n'a aucune raison d'être un ensemble algébrique réel. Ceci est simplement dû au fait que la topologie de Zariski et la topologie usuelle ne coïncident pas sur $\mathbb{R}^n$. Si on voulait adapter la notion de variété discriminante minimale, W_D serait en fait un ensemble semi-algébrique et non algébrique, difficile à caractériser simplement.

On se contentera donc de la notion de variété discriminante (non minimale) avec à l'esprit qu'il s'agit certainement de toutes façons d'un passage obligé, pour le moins algorithmiquement, puisqu'il n'existe quasiment pas d'algorithmes étudiant les zéros réels de systèmes semi-algébriques et même algébriques qui ne résolve pas l'équivalent complexe avant (implicitement ou explicitement). C'est flagrant pour ce qui concerne l'algorithme proposé dans le chapitre 3, un peu plus masqué dans le cas des polynômes en une variables ou l'on peut noter toutefois un lien direct entre les sous-résultants et les suites de Sturm. Même les algorithmes basés sur la règle de Descartes dépendent fortement de la structure complexe du problème (voir critères de terminaison dans le chapitre 2).

Pour se rapprocher d'un objet optimal il est naturel de considérer $W_D \cap \mathbb{R}^d$, mais quelques précautions doivent être prises :

- [A] Dans le cas ou $W_D = \overline{\Pi_U(\mathcal{C})}$, par définition, $\mathcal{C}_{|_{U=u}}$ est soit vide soit admet une infinité de solutions pour presque tout $u \in \Pi_U(\mathcal{C})$. Mais, dans ce dernier cas, $\mathcal{C}_{|_{U=u}} \cap \mathbb{R}^{n-d}$ et en particulier $\mathcal{S}_{|_{U=u}}$ peut tout à fait admettre un nombre fini de solutions, auquel cas $W_D \cap \mathbb{R}^d$ n'est pas une variété discriminante de $\mathcal{S}$ si on dérive la définition 5.8 dans le cas réel.
- [B] Dans le cas ou $W_D \subsetneqq \overline{\Pi_U(\mathcal{C})}$, il se peut que $\mathcal{C}_{|_{U=u}}$ soit non vide et fini pour tout $u \in \Pi_U(\mathcal{C}) \subset W_D$ mais que $\mathcal{C}_{|_{U=u}} \cap \mathbb{R}^{n-d}$ (et a fortiori, $\mathcal{S}_{|_{U=u}}$) soit vide pour tout $u \in (\Pi_U(\mathcal{C}) \cap \mathbb{R}^d) \setminus (W_D \cap \mathbb{R}^d)$, auquel cas, $W_D \cap \mathbb{R}^d$ n'est pas une variété discriminante de $\mathcal{C} \cap \mathbb{R}^n$ (et encore moins de $\mathcal{S}$) pour Π_U.

Dans tous les autres cas, $W_D \cap \mathbb{R}^d$ est une variété discriminante de $\mathcal{C} \cap \mathbb{R}^n$ pour Π_U, c'est à dire que si $\mathcal{U}$ est un ouvert connexe de $\Pi_U(\mathcal{C} \cap \mathbb{R}^n) \setminus (W_D \cap \mathbb{R}^d)$ alors $(\Pi_U^{-1}(\mathcal{U}) \bigcap \mathcal{C} \cap \mathbb{R}^n, \Pi_U)$ est un revêtement analytique de $\mathcal{U}$.

Les situations [A] et [B] ont pour cause une différence de dimension entre certaines composantes de $\mathcal{C}$ et leur intersection avec $\mathbb{R}^n$.

Dans la situation [A], il existe une composante $\mathcal{D} \subset \mathcal{C}$ de dimension $> \delta$ telle que sa projection soit dense dans $\Pi_U(\mathcal{C})$ et telle que la dimension de $\mathcal{D} \cap \mathbb{R}^n$ soit $\leq \delta$. Sauf si $\overline{\mathcal{C}} = \mathbb{C}^n$, on peut toujours trouver $Y \subset [U, X]$ tel que $U \subset Y$ et que la variété discriminante minimale W'_D de $\mathcal{C}$ relativement à Π_Y ne soit pas égale à $\overline{\Pi_Y(\mathcal{C})}$. On est alors ramenés à tester la situation [B].

Dans la situation [B], tout point $x \in \mathcal{C} \cap \mathbb{R}^n$ vérifie $\Pi_U(x) \in W_D \cap \mathbb{R}^d$. Plus précisément, tout point $x \in \mathcal{C} \cap \mathbb{R}^n$ est soit singulier (donc appartenant à O_c) soit élément de $\Pi_U^{-1}(O_\infty)$. Pour tester si on est dans cette situation ou non, il « suffit » de prendre un point u dans chaque composante connexe de $(\Pi_U(\mathcal{C}) \cap \mathbb{R}^d) \setminus (W_D \cap \mathbb{R}^d)$ et de résoudre $\mathcal{C}_{|_{U=u}}$ pour tester si $\mathcal{C}_{|_{U=u}} \cap \mathbb{R}^{n-d}$ est vide ou non. Si c'est le cas, $\mathcal{C} \cap \mathbb{R}^n \subset \Pi_U^{-1}(W_D) \cap \mathcal{C} \cap \mathbb{R}^d$ et on peut alors remplacer $\mathcal{C}$ par $\mathcal{C}' = \mathcal{C} \cap W_D$ pour étudier $\mathcal{C} \cap \mathbb{R}^n$. Notons que $\mathcal{C}'$ n'est pas forcément de dimension plus petite que $\mathcal{C}$ mais par contre $\mathcal{C}' \subsetneqq \mathcal{C}$ puisque l'on s'est débarrassé de toutes les composantes de dimension δ sont la projection était dense dans $\Pi_U(\mathcal{C})$.

On peut effectuer exactement la même construction en remplaçant dans ce qui précède $\mathcal{C} \cap \mathbb{R}^n$ par $\mathcal{S}$ pour obtenir le corollaire :

Corollaire 5.9. *On peut calculer un ensemble constructible $\mathcal{C}' \subset \mathcal{C}$ tel que $\mathcal{C} \cap \mathbb{R}^n \subset \mathcal{C}'$ (resp. $\mathcal{S} \subset \mathcal{C}'$) et tel que si W' est une variété discriminante de de $\mathcal{C}'$ pour Π_U, alors $W' \cap \mathbb{R}^d$ est une variété discriminante de $\mathcal{C} \cap \mathbb{R}^n$ (resp. $\mathcal{S}$) pour Π_U.*

5.4 Variétés discriminantes, résolution de systèmes et effectivité

En définissant un objet qui soit une variété algébrique, on se facilite grandement la tâche dans l'optique de fournir un algorithme. Encore faut-il non seulement pouvoir le calculer effectivement mais également l'utiliser pour résoudre les problèmes soumis.

Le but de cette section est simplement de spécifier un minimum d'outils nécessaires pour le calcul et l'utilisation d'une variété discriminante, l'aspect « efficace » sera abordé dans une section ultérieure.

5.4.1 Calcul effectif de variétés discriminantes

Une opération sans laquelle rien ne sera possible est le calcul de $\overline{\Pi_U(V)}$, V étant un ensemble algébrique. Supposant que l'on connaisse le système de générateurs d'un idéal $\mathcal{I}$ tel que $V = \boldsymbol{V}(\mathcal{I})$, calculer $\overline{\Pi_U(V)}$ peut être vu comme calculer l'idéal $\mathcal{I} \cap \mathbb{Q}[U]$ (puisque $\boldsymbol{V}(\mathcal{I} \cap \mathbb{Q}[U]) = \overline{\Pi_U(V)}$) ce qui peut s'effectuer en calculant une base de Gröbner de $\mathcal{I}$ pour un ordre monomial d'élimination (voir [34] ou [16] par exemple).

Si aucun des $f \in \mathcal{F}$ ne s'annule sur aucune composante irréductible de $\langle \mathcal{E} \rangle$, alors on peut prendre $\mathcal{I} = \langle p_1, ..., p_s \rangle$. Sinon, il faut enlever ces composantes, la dimension de $\boldsymbol{V}(\langle p_1, ..., p_s \rangle)$ pouvant, par exemple, être supérieure à celle de $\overline{\mathcal{C}}$. En remarquant que $\overline{\mathcal{C}} = \boldsymbol{V}(\mathcal{I})$, avec $\mathcal{I} = \langle \mathcal{E} \rangle$: $\left(\prod_{f \in \mathcal{F}} f \right)^{\infty}$, qui se réécrit $\mathcal{I} = \left(\langle \mathcal{E} \rangle + \langle T \prod_{f \in \mathcal{F}} f - 1 \rangle \right) \cap \mathbb{Q}[U, X]$, T étant une nouvelle variable, on se ramène au problème général d'élimination.

De la même manière, en posant $\mathcal{I}_{\Pi_U} = \mathcal{I} \cap \mathbb{Q}[U]$ ou encore $\mathcal{I}_f = (\mathcal{I} + \langle f \rangle) \cap \mathbb{Q}[U]$, on obtient $\overline{\Pi_U(\mathcal{C})} = \boldsymbol{V}(\mathcal{I}_{\Pi_U})$ et $\overline{O_{\mathcal{F}}} = \cup_{f \in \mathcal{F}} \boldsymbol{V}(\mathcal{I}_f)$ par de simples calculs d'élimination.

Le lemme 5.5 donne directement une solution algorithmique pour le calcul de O_∞ : on calcule idéal homogène $\mathcal{I}^h$ tel que $\boldsymbol{V}(\mathcal{I}^h) = \overline{C^{\mathbb{P}}}$, ce qui peut par exemple se faire en homogénéisant les polynômes d'une base de Gröbner de $\mathcal{I}$ calculée pour un ordre du degré [34], puis on intersecte avec $\mathcal{H}_\infty$, ce qui consiste à remplacer la variable d'homogénéisation utilisée pour le calcul de $\mathcal{I}^h$ par 0 dans les générateurs de $\mathcal{I}^h$ (notons $\mathcal{I}^h_{|h=0}$ ce résultat). Il suffit enfin de calculer $\mathcal{I}_\infty = \mathcal{I}^h_{|h=0} \cap \mathbb{Q}[U]$ pour obtenir le résultat, ce qui nous ramène au problème général de l'élimination.

Une remarque importante est que les méthodes permettant de calculer de $\mathcal{I}_{\Pi_U}$, $\mathcal{I}_\infty$, $\mathcal{I}_f$ s'appliquent quelque soit la structure de l'idéal $\mathcal{I}$ (qui peut être remplacé par n'importe quel autre idéal ayant les mêmes zéros).

Trois problèmes vont se poser pour les autres composantes :

- La non équi-dimensionalité de $\mathcal{I}$ et/ou de $\mathcal{I}_{\Pi_U}$;
- La non radicalité de $\mathcal{I}$ et/ou de $\mathcal{I}_{\Pi_U}$;
- La présence de composantes immergées;

Si on suppose que $\mathcal{I}$ est premier et que $\overline{\Pi_U(\mathcal{C})} = \mathbb{C}^d$, on contourne ces contraintes. En effet, $O_{\text{sd}} = O_{\text{sing}} = \emptyset$ et le calcul des points critiques « au sens large », peut se faire simplement puisque si $\mathcal{I}_c = \left(\mathcal{I} + \langle \text{Jac}_X^{n-d}(\mathcal{E}) \rangle \right) \cap \mathbb{Q}[U]$, alors $\boldsymbol{V}(\mathcal{I}_c) = \overline{O_c}$, avec les notations suivantes :

Notation 5.10. *Pour tout $\mathcal{G} = \{p_1, ..., p_s\} \subset \mathbb{Q}[Y_1, ..., Y_k]$ tout $Y' = [Y_{n_1}, ..., Y_{n_{k'}}] \subset [Y_1, ..., Y_k]$ et tout entier $m \leq k'$, on définit $\text{Jac}^m_{Y'}(\mathcal{G})$, l'ensemble des mineurs $m \times m$ de la matrice $\left[\frac{\partial p_i}{\partial Y_{n_j}} \right]_{i=1...s}^{j=1...k'}$.*

Même en supposant que $\overline{\Pi_U(\mathcal{C})} = \mathbb{C}^d$, on ne peut pas étendre cette caractérisation au cas d'idéaux généraux. Par exemple, si $\mathcal{E} = \{P^2(X)\}$, avec $P \neq 0$, alors $\boldsymbol{V}(\mathcal{I} + \langle \mathrm{Jac}_X^{n-d}(\mathcal{E})\rangle) = \boldsymbol{V}(\mathcal{I})$ et, en particulier, $\left(\mathcal{I} + \langle \mathrm{Jac}_X^{n-d}(\mathcal{E})\rangle\right) \cap \mathbb{Q}[U] = \mathcal{I}_{\Pi_U}$. On ne pourra donc pas remplacer $\overline{O_c}$ par $\boldsymbol{V}(\mathcal{I}_c)$ dans la définition d'une variété discriminante, même non minimale, puisque que le système admet, pour toutes les valeurs des paramètres, un nombre fini non nul de solutions complexes.

Par contre, on peut montrer facilement que cette définition convient dès lors que $\mathcal{I}$ est équidimensionnel et radical, toujours dans le cas ou $\overline{\Pi_U(\mathcal{C})} = \mathbb{C}^d$. Si on lève d'hypothèse $\overline{\Pi_U(\mathcal{C})} = \mathbb{C}^d$, on a toujours $O_{\mathrm{sd}} = \emptyset$, on peut éventuellement caractériser $\overline{O_c}$ simplement, mais $\mathcal{I}_{\Pi_U}$ peut être quelconque et on se retrouve confrontés au même type de problème pour caractériser O_{sing} que ceux rencontrés pour O_c: l'ensemble $\boldsymbol{V}(\mathcal{I}_{\Pi_U} + \langle \mathrm{Jac}_U^{\delta}(\mathcal{I}_{\Pi_\mathcal{U}})\rangle)$ peut ne pas contenir O_{sing}, ne pas être contenu dans O_{sing} ou être simplement trop gros pour permettre le calcul d'une variété discriminante, même non minimale.

Si on ne se préoccupe que du caractère effectif (et non efficace) du calcul de variété discriminante, une solution est de calculer et décomposer $\sqrt{\mathcal{I}}$ (et au besoin $\mathcal{I}_{\Pi_U}$) comme intersection d'idéaux premiers $\sqrt{\mathcal{I}} = \cap_{i=1\ldots l}\, \mathcal{P}_i$: par exemple les algorithmes présentés dans [16], [53] ou encore [41] permettent de calculer tous les $\mathcal{P}_i$.

Comme nous l'avons vu ci-dessus, le calcul d'une variété discriminante est direct dès lors que $\mathcal{I}$ est premier. Si on décompose $\sqrt{\mathcal{I}}$ en intersection d'idéaux premiers $\mathcal{P}_i$, un réflexe est alors de calculer des variétés discriminantes (de préférence minimales) des $\boldsymbol{V}(\mathcal{P}_i)$ pour la projection Π_U, le rapport avec une variété discriminante de $\mathcal{I}$ pouvant être établi comme suit :

- Soit $\dim(\mathcal{P}_i) < \dim(\mathcal{P}_i \cap \mathbb{Q}[\mathcal{U}])$, et, dans ce cas, $\boldsymbol{V}(\mathcal{P}_i \cap \mathbb{Q}[U]) \subset \overline{O_\infty}$
- Soit $\dim(\mathcal{P}_i) = \dim(\mathcal{P}_i \cap \mathbb{Q}[\mathcal{U}]) < \delta$ et, dans ce cas, $\boldsymbol{V}(\mathcal{P}_i \cap \mathbb{Q}[U]) \subset \overline{O_{\mathrm{sd}}}$; En notant $\mathcal{I}^i_{\mathrm{sd}} = \mathcal{P}_i \cap \mathbb{Q}[\mathcal{U}]$, il vient de plus que $\overline{O_{\mathrm{sd}}} = \cup_{i=1\ldots l}\, \boldsymbol{V}(\mathcal{I}^i_{\mathrm{sd}})$.
- Dans tous les autres cas, $\dim(\mathcal{P}_i) = \dim(\mathcal{P}_i \cap \mathbb{Q}[U]) = \delta$ et, en particulier,
  - $\mathcal{I}^i_c = \left(\mathcal{P}_i + \langle \mathrm{Jac}_X^{n-d}(\mathcal{P}_i)\rangle\right) \cap \mathbb{Q}[U] \subset \overline{O_c}$
  - $\overline{O_c \setminus (O_{\mathrm{sd}} \cup O_\infty)} = \cup_{i=1\ldots l}\, \boldsymbol{V}(\mathcal{I}^i_c) \cup_{i\neq j, i,j=1\ldots l} \boldsymbol{V}((\mathcal{P}_i + \mathcal{P}_j) \cap \mathbb{Q}[U])$

Pour le calcul de $\overline{O_{\mathrm{sing}}}$, lorsque cet ensemble n'est pas trivialement vide, on peut d'emblée ne pas étudier les $\mathcal{P}_i$ vérifiant les hypothèses des deux premiers points et se concentrer sur les points singuliers de la variété constituée des zéros de l'idéal $\mathcal{I'}_{\Pi_U} = \cap_{i=1\ldots l'} \mathcal{P}_{n_i} \cap \mathbb{Q}[U]$ ou les $\mathcal{P}_{n_i}$ vérifient $\dim(\mathcal{P}_{n_i}) = \dim(\mathcal{P}_{n_i} \cap \mathbb{Q}[U]) = \delta$: les autres points singuliers de $\mathcal{I}_{\Pi_U}$ seront dans $O_\infty \cup O_{\mathrm{sd}}$. Comme précédemment, on peut supposer connue une décomposition de $\sqrt{\mathcal{I'}_{\Pi_U}}$ comme intersection d'idéaux premiers $\sqrt{\mathcal{I'}_{\Pi_U}} = \cap_{i=1\ldots l'} \mathcal{P'}_i$ et remarquer que :

- $\mathcal{I}^i_{\mathrm{sing}} = \left(\mathcal{P}'_i + \langle \mathrm{Jac}_U^{d-\delta}(\mathcal{P}'_i)\rangle\right) \subset \overline{O_{\mathrm{sing}}}$
- $\overline{O_{\mathrm{sing}} \setminus (O_{\mathrm{sd}} \cup O_\infty)} = \cup_{i=1\ldots l'}\, \boldsymbol{V}(\mathcal{I}^i_{\mathrm{sing}}) \cup_{i\neq j, i,j=1\ldots l'} \boldsymbol{V}(\mathcal{P}'_i + \mathcal{P}'_j)$

5.4.2 Utilisation de variétés discriminantes pour la résolution de systèmes paramétrés

On a montré dans la section précédente que l'on peut, en principe, toujours calculer la variété discriminante minimale de $\mathcal{C}$ pour Π_U ou encore de $\mathcal{S}$ pour Π_U modulo l'adaptation au cas réel proposée dans la section 5.3 de ce chapitre. Cette courte partie est simplement destinée à montrer comment elle peut être utilisée dans le cas réel (le cas complexe sortant du cadre de ce document) ou plutôt comment l'exploiter en pratique pour « résoudre » le système $\mathcal{S}$.

Pour simplifier, on suppose que W_D est une variété discriminante de $\mathcal{C}$ ainsi que de $\mathcal{C} \cap \mathbb{R}^n$ et de $\mathcal{S} \cap \mathbb{R}^n$ pour Π_U, situation à laquelle on peut toujours se ramener d'après la section 5.3. On note $\mathcal{U}_1, \ldots, \mathcal{U}_l$ les composantes connexes de $\Pi_U(\mathcal{C}) \cap \mathbb{R}^d \setminus W_D \cap \mathbb{R}^d$.

La propriété principale d'une variété discriminante est que pour tout $\mathcal{U}_i$ (resp. tel que $\Pi_U^{-1}(\mathcal{U}_i) \cap \mathcal{S} \neq \emptyset$), $(\Pi_U^{-1}(\mathcal{U}_i) \cap \mathcal{C} \cap \mathbb{R}^n, \Pi_U)$ (resp. $(\Pi_U^{-1}(\mathcal{U}_i) \cap \mathcal{S}, \Pi_U)$) est un revêtement analytique de $\mathcal{U}_i$.

En particulier, pour i fixé, le nombre de solutions de $\mathcal{C}_{|U=u} \cap \mathbb{R}^{n-d}$ (resp. $\mathcal{S}_{|U=u}$) est constant pour tout $u \in \mathcal{U}_i$, ce nombre pouvant être calculé en résolvant $\mathcal{C}_{|U=u_0} \cap \mathbb{R}^{n-d}$ (resp. $\mathcal{S}_{|U=u_0}$) en choisissant n'importe quel $u_0 \in \mathcal{U}_i$. De plus, la propriété de revêtement analytique assure que les feuilles $\Pi_U^{-1}(\mathcal{U}_i) \cap \mathcal{C} \cap \mathbb{R}^n$ (resp. $\Pi_U^{-1}(\mathcal{U}_i) \cap \mathcal{S}$) ne s'intersectent pas, sont non singulières et sont localement difféomorphes à $\mathcal{U}_i$: à défaut de produire directement une formule littérale des solutions (ce qui est, comme nous l'avons vu, inutile en général au regard des questions qui se posent, en pratique, sur de tels systèmes) la connaissance des $\mathcal{U}_i$ donne donc toute la structure des solutions pour des paramètres non contenus dans W_D.

Pour résumer, les variétés discriminantes (minimales ou non) permettent de caractériser directement les solutions « génériques » d'un système $\mathcal{C}$, $\mathcal{C} \cap \mathbb{R}^n$ ou encore $\mathcal{S}$, le terme « générique » étant ici complètement spécifié puisqu'il s'agit des solutions pour les valeurs des paramètres contenues dans $\Pi_U(\mathcal{C}) \setminus W_D$ ou $\Pi_U(\mathcal{C}) \cap \mathbb{R}^d \setminus W_D \cap \mathbb{R}^d$.

Une remarque est, qu'en principe en tout cas, l'information sur les solutions « non génériques » n'est pas perdue, puisque l'on peut poursuivre l'étude en étudiant simplement $\mathcal{C} \cap W_D$ (ou $\mathcal{C} \cap W_D \cap \mathbb{R}^n$ ou encore $\mathcal{S} \cap W_D$ selon le problème initial). La récursion s'arrêtera alors lorsque le système admet soit aucune solution soit une infinité de solutions pour tout paramètre de la variété discriminante.

A vrai dire, l'étude de toutes les solutions du système n'a généralement aucun intérêt en pratique, aussi nous nous concentrons essentiellement sur l'étude des « solutions génériques ».

Que ce soit pour la stratégie développée dans la section 5.3 pour construire une variété discriminante dans le cas réel ou encore calculer le nombre de solutions réelles en fonction des paramètres, un algorithme calculant un point par composante connexe de $\Pi_U(\mathcal{C}) \cap \mathbb{R}^d \setminus W_D \cap \mathbb{R}^d$ représente une boite noire essentielle. On peut estimer ce problème résolu de manière effective puisque qu'il existe au moins deux types d'algorithmes permettant de fournir ce type de résultat :

- les méthodes de points critiques (voir Chapitre 4) permettent de calculer au moins un point par composante connexe de $\Pi_U(\mathcal{C}) \cap \mathbb{R}^d \setminus W_D \cap \mathbb{R}^d$. Notons que l'on aura pas de description des $\mathcal{U}_i$, le couplage d'une telle boite noire avec le calcul d'une variété discriminante ne permettra donc que d'obtenir des résultats partiels comme le nombre maximum/minimum de racines.
- une décomposition cylindrique algébrique (CAD) adaptée aux polynômes définissant $\mathcal{C}$ et W_D permettra de calculer une description cylindrique des $\mathcal{U}_i$ exploitable en pratique et de calculer un point dans chacune de ces cellules. La conjonction de ces deux objets donne alors une description complète des solutions du système.

Ces éléments sont proposés ici en dehors de toute considération de complexité ou d'efficacité. En particulier, on peut tout à fait calculer directement une CAD adaptée aux polynômes de $\mathcal{C}$ et obtenir un résultat ayant les mêmes spécifications. Dans cette logique, la variété discriminante peut être vue comme le remplacement des $n-d$ premières étapes de la phase de projection d'un algorithme calculant une CAD. Nous verrons plus loin que ce remplacement est avantageux autant en pratique (efficacité) qu'en théorie (complexité), mais on peut déjà s'en convaincre en remarquant simplement que le calcul de variété discriminante tire profit de la présence d'égalités en décomposant $\boldsymbol{V}(\mathcal{E})$ en régions ou les polynômes de $\mathcal{F}$ sont de signes constants alors que la CAD décompose l'espace ambiant en régions ou les polynômes de $\mathcal{F} \cup \mathcal{E}$ sont de signes constants, engendrant mécaniquement beaucoup plus de cas à distinguer. Cet argument ne tient plus pour l'étude de $\Pi_U(\mathcal{C}) \cap \mathbb{R}^d \setminus W_D \cap \mathbb{R}^d$, en particulier lorsque $\overline{\Pi_U(\mathcal{C})} = \mathbb{C}^d$ et la CAD est l'unique option crédible pour obtenir un résultat complet, c'est à dire fournissant une description exploitable des $\mathcal{U}_i$.

5.5 Le cas des systèmes « bien formulés »

L'introduction de ce chapitre rappelle que les systèmes apparaissant dans les applications ont en général quelques propriétés remarquables :

- $\#\mathcal{E} = n - d$

- $\langle \mathcal{E}_{|_{U=u}} \rangle$ est zéro-dimensionnel et radical pour presque toutes les valeurs de $u \in \mathbb{C}^d$
- $\mathcal{C}|_{U=u}$ admet un nombre fini (non nul) de solutions pour presque toutes les valeurs de $u \in \mathbb{C}^d$

Bien que le but ultime soit de produire un algorithme « universel », un objectif affiché est de résoudre efficacement et en priorité ces systèmes (les plus courants) que nous qualifierons de systèmes « bien formulés » et que l'on peut caractériser par :

Notation 5.11. *[86],[32] $\mathcal{C}$ ou $\mathcal{S}$ est « bien formulé » si :*

- $\#\mathcal{E} = n - d$
- *$\langle \mathcal{E} \rangle$ ainsi que tout idéal $\mathcal{I}$ tel que $\boldsymbol{V}(\mathcal{I}) = \bar{\mathcal{C}}$ sont zéro-dimensionnels et radicaux dans $\mathbb{Q}(U)[X]$.*

Les deux hypothèses résument que l'on peut résoudre numériquement la plupart des spécialisations par une méthode simple et standard telle que la méthode de Newton, ce qui est raisonnable car, en général, l'ingénieur ou le chercheur modélise le plus souvent son problème de sorte à ce que ces deux conditions soient remplies.

Par contre, le système final est obtenu le plus souvent après quelques changements de variables et réductions de dénominateurs, entraînant l'apparition de composantes primaires non premières de toutes les dimensions (dans l'application « robots cuspidaux » du chapitre 6, on pourra par exemple voir l'effet d'un changement de variable introduisant la tangente de l'angle moitié à la place des sinus et cosinus, en particulier sur la dimension du système). Souvent, les inégalités ne sont présentes que pour corriger les artefacts de mises en équations (apparition de composantes parasites liées aux réduction des dénominateurs, domaine d'application des changements de variables, etc.) ou délimiter un pavé de $\mathbb{R}^n$ pour les zéros de $\langle \mathcal{E} \rangle$. En pratique, ceci se traduit, en général, par le fait qu'il n'existe pas de composante irréductible de $\mathcal{F}$ sur laquelle un polynôme de $\mathcal{F}$ soit toujours nul.

En revanche, il faut noter que ces conditions n'impliquent pas que $\langle \mathcal{E} \rangle$ ou $\mathcal{I}$ soit équi-dimensionnel ni de dimension d, ni radical dans $\mathbb{Q}[U, X]$, ce qui est d'ailleurs dommage au regard de la section 5.4.

Pour un système « bien formulé », $\overline{\Pi_U(\mathcal{C})} = \mathbb{C}^d$ et, en particulier, $O_{\text{sing}} = \emptyset$, puisque $\mathcal{I}$ est zéro-dimensionnel dans $\mathbb{Q}(U)[X]$. Comme, de plus, $\#\mathcal{E} = n - d$, on a également $O_{\text{sd}} = \emptyset$.

Bien que la variété $\boldsymbol{V}(\langle \mathcal{E} \rangle)$, n'ait aucune composante irréductible de dimension $< d$, on ne peut toutefois pas supposer que les idéaux $\langle \mathcal{E} \rangle$ et $\mathcal{I}$ n'aient aucune composante primaire de dimension $< d$: il peut très bien y avoir des composantes immergées qui compliquent en particulier la caractérisation de O_c. Même en l'absence de composantes immergées, on ne peut pas non plus supposer que $\langle \mathcal{E} \rangle$ ou $\mathcal{I}$ soit radical. Si on suppose raisonnable de calculer un idéal d'élimination, le calcul le plus délicat est alors celui de O_c puisqu'il requiert a priori le calcul et la décomposition de $\sqrt{\mathcal{I}}$, pré-requis connu pour être d'un coût prohibitif, en tout cas avec les algorithmes de l'état de l'art.

Pour bien voir les optimisations possibles dans le cas de ce type de systèmes, il faut regarder la structure des idéaux $\langle \mathcal{E} \rangle$ ou de manière presque équivalente, $\mathcal{I}$ et en particulier l'allure de leurs décompositions primaires.

Notation 5.12. *On note $\langle \mathcal{E} \rangle = \cap_{i=1\ldots i_1} \mathcal{Q}_i \cap_{i=1\ldots i_2} \mathcal{Q}'_i$ une décomposition primaire minimale de $\langle \mathcal{E} \rangle$ telle que $\forall i = 1 \ldots i_1, \dim(\mathcal{Q}_i) = \dim(\mathcal{Q}_i \cap \mathbb{Q}[U]) = \delta$. Dans le cas d'un système « bien formulé », $\delta = d$ et, $\forall i = 1 \ldots i_2, \dim(\mathcal{Q}'_i) < \delta$. On pose alors $\mathcal{I}^{\text{ec}} = \cap_{i=1\ldots i_1} \mathcal{Q}_i$ (l'idéal obtenu par extension puis contraction de $\mathcal{I}$).*

Avec ces notations, on peut faire les deux remarques suivantes [86] :

- Le fait que $\langle \mathcal{E} \rangle$ soit radical dans $\mathbb{Q}(U)[X]$, implique que $\mathcal{I}^{\text{ec}}$ est équi-dimensionnel et radical et donc que les points critiques « au sens large » de Π_U en restriction à $\boldsymbol{V}(\mathcal{I}^{\text{ec}})$ sont exactement les zéros de l'idéal $\mathcal{I}^{\text{ec}} + \text{Jac}_X^{n-d}(\mathcal{E})$.
- L'autre remarque est que les idéaux $\mathcal{I} + \text{Jac}_X^{n-d}(\mathcal{I})$ et $\mathcal{I} + \text{Jac}_X^{n-d}(\mathcal{E})$ sont égaux sous l'hypothèse que le système $\mathcal{C}$ est « bien formulé ».

On pourrait bien sur calculer $\mathcal{I}^{\text{ec}}$, ce qui représenterait déjà un progrès , en pratique, comparé au calcul et à la décomposition de $\mathcal{I}$, en remarquant de plus qu'il s'agit d'une opération qui sera de toutes façons effectuée si on utilise les algorithmes de décomposition les plus connus. En fait, un tel calcul est inutile si on assure l'absence de composantes immergées, puisque, dans ce cas, $\boldsymbol{V}(\mathcal{I}^{\text{ec}} \cap \mathbb{Q}[U]) = \overline{O_c}$. Le lemme suivant montre que cette hypothèse peut systématiquement être levée dès lors que $\#\mathcal{E} = n - d$:

Lemme 5.13. ***FR**-[86] Si on suppose que $\#\mathcal{E} = n - \delta$, alors, en reprenant les notations 5.12, tout $\mathcal{Q}'_i$ tel qu'il existe $\mathcal{Q}_j$ tel que $\sqrt{\mathcal{Q}_j} \subset \sqrt{\mathcal{Q}'_i}$ vérifie $\boldsymbol{V}(\mathcal{Q}'_i \cap \mathbb{Q}[U]) \subset O_\infty$.*

La combinaison des remarques liés au cas particulier des systèmes « bien posés » et de ce lemme donne alors le résultat suivant :

Lemme 5.14. ***FR**-[86] Si on suppose $\mathcal{C}$ « bien formulé » et si $\mathcal{I} \subset \mathbb{Q}[U, X]$ est un idéal tel que $\boldsymbol{V}(\mathcal{I}) = \overline{\mathcal{C}}$, alors $\boldsymbol{V}(\left(\mathcal{I} + \text{Jac}_X^{n-d}(\mathcal{E})\right) \cap \mathbb{Q}[U]) \cup O_\infty \cup O_\mathcal{F}$ est une variété discriminante minimale de $\mathcal{C}$ pour Π_U.*

On a ainsi montré que, pour si $\mathcal{C}$ est « bien formulé », le calcul de la variété discriminante minimale de $\mathcal{C}$ pour Π_U peut s'effectuer avec un nombre minimal d'opérations simples sur les idéaux, essentiellement l'élimination de variables.

En fait, la plupart des outils de base utilisés pour démontrer les résultats utiles pour le cas des systèmes « bien formulés » permettent de mettre en évidence quelques propriétés utiles pour la résolution de systèmes plus généraux :

Lemme 5.15. ***FR**-[86] Si $\dim(\boldsymbol{V}(\left(\mathcal{I} + \text{Jac}_X^{n-d}(\mathcal{E})\right) \cap \mathbb{Q}[U])) < \delta$, alors $\boldsymbol{V}(\left(\mathcal{I} + \text{Jac}_X^{n-\delta}(\mathcal{E})\right) \cap \mathbb{Q}[U]) \cup O_\infty \cup O_\mathcal{F} \cup O_{\text{sd}} \cup O_{\text{sing}}$ est une variété discriminante de $\mathcal{C}$ pour Π_U. Si, de plus, $\#\mathcal{E} = n - \delta$, elle est minimale.*

De la même façon, si $\dim(\boldsymbol{V}(\mathcal{I}_{\Pi_U} + \text{Jac}_U^{d-\delta}(\mathcal{I}_{\Pi_U}))) < \delta$, alors $\boldsymbol{V}(\mathcal{I}_{\Pi_U} + \text{Jac}_U^{d-\delta}(\mathcal{I}_{\Pi_U})) \cup O_\infty \cup O_\mathcal{F} \cup O_{\text{sd}} \cup O_c$ est une variété discriminante de $\mathcal{C}$ pour Π_U.

5.6 Un algorithme général pour le calcul de variété discriminante

L'objectif de cette section est de proposer un algorithme général mais qui soit le plus efficace possible sur les systèmes « bien formulés », puisque ceux-ci représentent la majorité des « challenges » soumis. Ceci suppose, en particulier, être capable de détecter cette catégorie de problèmes le plus tôt possible et d'avoir une stratégie agressive pour les résoudre.

Comme on a déjà pu s'en rendre compte, dès lors que l'on connaît $\mathcal{I}$ tel que $\boldsymbol{V}(\mathcal{I}) = \overline{\mathcal{C}}$, une partie des calculs ne dépend a priori pas de la structure des idéaux manipulés ni des propriétés remarquables de $\mathcal{C}$: c'est le cas par exemple de O_∞, $O_\mathcal{F}$, ou encore $\overline{\Pi_U(\mathcal{C})}$. Par contre, les calculs nécessaires pour obtenir O_c, O_{sing} ou encore O_{sd} sont assurément de complexités variables en fonction des propriétés de $\mathcal{I}$. Ceci induit un ordre naturel dans lequel effectuer les calculs : en priorité les composantes qui peuvent être calculées de façon universelle, ensuite les autres, avec comme comme objectif intermédiaire de détecter le plus tôt possible les propriétés remarquables permettant d'optimiser les calculs (par exemple détection des systèmes « bien formulés »).

Le cas de O_{sd} est moins sensible car, pour la plupart des applications, les composantes de petite dimension peuvent être simplement ignorées, puisque provenant, la plupart du temps, d'artefacts de la mise en équation. Une autre raison est que les paramètres sont bien souvent liés à des grandeurs physiques qui ne peuvent être mesurées ou évaluées de manière exacte : il ne fait aucun sens de les considérer liées par une relation algébrique.

Plus généralement, l'algorithme complet qui sera proposé au final, tient compte des écarts que l'utilisateur peut tolérer. On pourra ainsi préciser une sortie minimale :

- La variété discriminante minimale;
- Une variété discriminante non nécessairement minimale;
- Une variété discriminante ne tenant pas compte des composantes de petite dimension : on l'appellera « variété discriminante partielle ».

5.6.1 Les algorithmes externes

Au regard des récents progrès et en particulier à la puissance des algorithmes F_4 [44] et F_5 [50], il est naturel que nos boites de calcul de base utilisent massivement des bases de Gröbner. Comme déjà évoqué dans le chapitre 3, une base de Gröbner est définie de manière unique pour un ordre monomial fixé, la rapidité du calcul d'un tel objet dépendant fortement de cet ordre, que ce soit en pratique ou en théorie (complexité). Pour les mêmes raisons que dans le cas des systèmes zéro-dimensionnels, on privilégie les ordres dits « du degré » et principalement de type DRL (Degree Reverse Lexicographique).

Comme il est normal de considérer séparément les paramètres U et les indéterminées X, on rappelle la notion d'ordre produit qui nous sera utile tout au long de cette section :

Notation 5.16. *Soit $Y = [U_1, ..., U_d, X_{d+1}, ..., X_n]$. Si $<_U$ (resp. $<_X$) est un ordre monomial pour les monômes ne dépendant que de U (resp. X), alors $<_{U,X}$ sera l'ordre monomial produit tel que $U_i <_{U,X} X_j$ pour tout $U_i \in U$ et tout $X_j \in X$.*

Sauf mention contraire, on supposera par défaut que $<_X$ et $<_U$ sont deux ordres DRL.

Pour tout polynôme $g \in \mathbb{Q}[U, X]$, $\mathrm{LM}_{<_X}(g)$ (resp. $\mathrm{LC}_{<_X}(g)$), désignera le monôme de tête (resp. le coefficient de tête) de g pour $<_X$, considérant alors g comme un polynôme en les variables X dont les coefficients sont dans $\mathbb{Q}[U]$ ($\mathrm{LC}_{<_X}(g)$ sera donc un polynôme en les variables U).

Les deux opérations les plus utilisées seront l'élimination de variables et la saturation d'un idéal par un polynôme ou un produit de polynômes (ce seront d'ailleurs les seules nécessaires pour la résolution des systèmes « bien formulés »):

Théorème 5.17. *[34],[16] Si G est une base de Gröbner d'un idéal $\mathcal{I} \subset \mathbb{Q}[U, X]$ pour n'importe quel ordre produit $<_{U,X}$, alors $G \bigcap \mathbb{Q}[U]$ est une base de Gröbner de $\mathcal{I} \bigcap \mathbb{Q}[U]$ w.r.t. $<_U$.*

Si T est une nouvelle indéterminée algébriquement indépendante de U et X, alors, pour tout $f \in \mathbb{Q}[U, X]$, $\overline{\boldsymbol{V}(\mathcal{I}) \setminus \boldsymbol{V}(f)} = \boldsymbol{V}((\mathcal{I} + \langle Tf - 1\rangle) \bigcap \mathbb{Q}[U, X])$.

Si $G' \subset \mathbb{Q}[U, X, T]$ est une base de Gröbner de $\mathcal{I} + \langle Tf - 1\rangle$ pour $<_{(U,X),T}$ alors $G' \bigcap \mathbb{Q}[U, T]$ est une base de Gröbner de $\mathcal{I}$: $f^\infty := (\mathcal{I} + \langle Tf - 1\rangle) \bigcap \mathbb{Q}[U, X]$ pour $<_{(U,X)}$. La variété $\overline{\boldsymbol{V}(\mathcal{I}) \setminus \boldsymbol{V}(f)}$ (resp. l'idéal $\mathcal{I}$: f^∞) sont généralement appelées saturations de $\boldsymbol{V}(I)$ (resp. $\mathcal{I}$) par f.

Bien que de nombreuses optimisations soient possibles dans le cadre du calcul d'une variété discriminante, nous utilisons, en pratique, des algorithmes généraux tels que ceux présents dans [6] ou [16], pour décomposer $\mathcal{I}$ lorsque c'est nécessaire, un objectif étant de toutes façons d'éviter à tout prix ce genre de calcul. En particulier, la préférence ira aux méthodes les plus paresseuses [6]. En effet, le travail concernant le cas des systèmes les plus généraux est actuellement en cours et les stratégies possibles de calcul plus adaptées ne sont pas encore bien cernées (voir le chapitre conclusions et perspectives de ce document).

On peut toutefois déjà donner donner les spécifications de ce que serait fonction de décomposition adaptée au regard des observations effectuées dans **FR**-[86], et en remarquant de plus qu'il est possible d'en proposer une implantation (inefficace) avec les outils de l'état de l'art :

Algorithm Decompose

- Input : $\mathcal{I} \subset \mathbb{Q}[U, X]$ an ideal, Π_U.
- Output : $\mathcal{I}_{\mathrm{main}}$ and $\mathcal{I}_{\mathrm{small}}$ such that :
  - $\boldsymbol{V}(\mathcal{I}_{\mathrm{small}}) \cap \boldsymbol{V}(\mathcal{I}_{\mathrm{main}}) = \emptyset$;
  - $\Pi_U(\boldsymbol{V}(\mathcal{I})) = \boldsymbol{V}(\mathcal{I}_{\mathrm{small}} \cap \mathbb{Q}[U])$;
  - $\dim(\mathcal{I}_{\mathrm{small}}) = \dim(\mathcal{I}_{\mathrm{small}} \cap \mathbb{Q}[U])$
  - $\mathcal{I}_{\mathrm{main}}$ is equidimensional, radical and such that $\dim(\mathcal{I}_{\mathrm{main}}) = \dim(\mathcal{I}_{\mathrm{main}} \cap \mathbb{Q}[U]) = \delta$;

Si on applique l'algorithme DECOMPOSE ainsi spécifié à notre ideal $\mathcal{I}$, on obtient alors :

- $O_\infty \cup O_\mathcal{F} \cup \boldsymbol{V}(\mathcal{I}_{\text{main}} + \text{Jac}_X^{n-\delta}(\mathcal{E})) \cup \boldsymbol{V}(\mathcal{I}_{\text{small}} \cap \mathbb{Q}[U]) \cup O_{\text{sing}}$ est une variété discriminante minimale de $\mathcal{C}$ pour Π_U;

Si on l'applique sur $\mathcal{I}_{\Pi_U}$, en indiquant que la fonction de projection est l'identité, on obtient $\mathcal{I}_{\Pi_U} = \mathcal{I}_{\text{small}} + \mathcal{I}_{\text{main}}$

5.6.2 Calculs inconditionnels

Comme application relativement directe des résultats rappelés dans la section précédente, l'algorithme qui suit « calcule » un idéal $\mathcal{I}$ tel que $\boldsymbol{V}(\mathcal{I}) = \overline{\mathcal{C}}$ ainsi que $\overline{\Pi_U(\mathcal{C})}$ et $\overline{O_\mathcal{F}}$. On notera que le plus grand soin est pris pour limiter le nombre de calculs de saturations pour obtenir $\overline{O_\mathcal{F}}$. En effet, il n'est nécessaire de saturer $\langle \mathcal{E} \rangle$ par les polynômes de $\mathcal{F}$ que si l'un des $f \in \mathcal{F}$ s'annule sur une composante irréductible $\mathcal{D} \subset \boldsymbol{V}(\langle \mathcal{E} \rangle)$ telle que $\overline{\Pi_U(\mathcal{D})} = \overline{\Pi_U(\mathcal{C})}$. Dans tous les autres cas, soit la saturation est théoriquement inutile (intersection transverse de $f = 0$ avec $\boldsymbol{V}(\langle \mathcal{E} \rangle)$), soit elle est inutile en pratique (l'intersection de $f = 0$ et de $\boldsymbol{V}(\langle \mathcal{E} \rangle)$ se projette, par Π_U, dans $O_{\text{sd}} \cup O_\infty$).

Algorithm PREPROCESSING

- **Input** : $\mathcal{E}, \mathcal{F}, U, X$
- **Output** : $\delta, G, G_\Pi, G_\mathcal{F}$ such that
  - G is a reduced Gröbner basis for $<_{U,X}$ such that $\mathcal{C} \cap \Pi_U^{-1}(\overline{\Pi_U(\mathcal{C})} \setminus O_\mathcal{F}) = \boldsymbol{V}(\langle G \rangle) \cap \Pi_U^{-1}(\overline{\Pi_U(\mathcal{C})} \setminus O_\mathcal{F})$;
  - G_Π, $G_\mathcal{F}$ are reduced Gröbner bases for $<_U$ such that $\boldsymbol{V}(\langle G_\Pi \rangle) = \overline{\Pi_U(\mathcal{C})}$ and $\boldsymbol{V}(\langle G_\mathcal{F} \rangle) = \overline{O_\mathcal{F}}$;
- **Begin**
  - **Compute** $G_\mathcal{E}$ the reduced Gröbner basis of $\mathcal{E}$ for $<_{U,X}$
  - **Deduce** $G_{\mathcal{E},U} = G_\mathcal{E} \cap \mathbb{Q}[U]$
  - **Compute** $d_{\mathcal{E},U}$, the dimension of $G_{\mathcal{E},U}$
  - **Compute** $G_{\mathcal{E}\cap\mathcal{F}}$, the reduced Gröbner basis of $\mathcal{E} \cup \{\prod_{i=1}^s f_i\}$ for $<_{U,X}$
  - **if** $(G_{\mathcal{E}\cap\mathcal{F}} = G_\mathcal{E})$ **then return** $(\delta = -1,\ G = \{1\},\ G_\Pi = \{1\},\ G_\mathcal{F} = \{1\})$
  - **else**
    - **Deduce** $G_{\mathcal{E}\cap\mathcal{F},U} = G_{\mathcal{E}\cap\mathcal{F}} \cap \mathbb{Q}[U]$
    - **Compute** $d_{\mathcal{E}\cap\mathcal{F},U}$, the dimension of $G_{\mathcal{E}\cap\mathcal{F},U}$
    - **if** $(d_{\mathcal{E},U} = d_{\mathcal{E}\cap\mathcal{F},U})$ **then**
      - **Compute** $G_{\mathcal{E},T\mathcal{F}}$, the reduced Gröbner basis of $\mathcal{E} \cup \{T(\prod_{i=1}^l f_i) - 1\}$ for $<_{T,(U,X)}$
      - **Deduce** $G_{\mathcal{E},\mathcal{F}} = G_{\mathcal{E},T\mathcal{F}} \cap \mathbb{Q}[X, U]$
      - **return**(PREPROCESSING$(G_{\mathcal{E},\mathcal{F}}, \mathcal{F}, U, X)$)
    - **else return**$(\ \delta = d_{\mathcal{E},U},\ G = G_\mathcal{E},\ G_\Pi = G_{\mathcal{E},U},\ G_\mathcal{F} = G_{\mathcal{E}\cap\mathcal{F},U})$
- **End**

La deuxième composante pouvant se calculer relativement directement, en tout cas sans hypothèse particulière sur $\mathcal{I}$ (ici G), est O_∞. Le résultat suivant montre que si G est une base de Gröbner de $\mathcal{I}$, calculée pour une ordre produit de deux ordres DRL, alors on peut « lire » O_∞ sur G sans autre calcul :

Théorème 5.18. ***FR**-[86] Soit G une base de Gröbner réduite, pour un ordre produit de deux ordres DRL $<_{U,X}$, d'un idéal $\mathcal{I}$ tel que $\boldsymbol{V}(I) = \bar{\mathcal{C}}$. Si on pose $\mathcal{E}_i^\infty = \{\mathrm{LM}_{<_X}(g) \mid g \in G, \exists m \geq 0, \mathrm{LM}_{<_X}(g) = X_i^m\}$, et $\mathcal{E}_0 = G \bigcap \mathbb{Q}[U]$, Alors :*

- *$\mathcal{E}_0$ est une base de Gröbner de $\mathcal{I} \bigcap \mathbb{Q}[U]$ pour l'ordre $<_U$ et $\mathcal{E}_0 \subset \mathcal{E}_i^\infty$ pour $i = d+1...n$;*
- *$\mathcal{E}_i^\infty$ est une base de Gröbner d'un certain idéal $I_i^\infty \subset \mathbb{Q}[U]$ pour l'ordre $<_U$;*
- $O_\infty = \bigcup_{i=d+1}^{n} \boldsymbol{V}(I_i^\infty)$.

On peut déjà remarquer que ce résultat donne un moyen de détecter partiellement si un problème est « bien formulé » : si $\mathcal{E}_0$ n'est pas vide, on saura que $\overline{\Pi_U(\mathcal{C})} \neq \mathbb{C}^d$ et si l'un des I_i^∞ est l'idéal nul, on saura alors que $\mathcal{C}_{|U=u}$ admet une infinité de solutions pour presque tout $u \in \Pi_U(\mathcal{C})$ (dans les deux cas, $\mathcal{I}$ n'est pas zéro-dimensionnel dans $\mathbb{Q}(U)[X]$). L'algorithme suivant décrit exactement le calcul effectué en pratique :

Algorithm PropernessDefects

- **Input** : $G_{U,X}$,U,X where $G_{U,X}$ a reduced Gröbner basis w.r.t $<_{U,X}$ where $<_U$ and $<_X$ are Degree Reverse Lexicographic orderings;
- **Output** : $G_i^\infty, i = d+1...n$ such that
  - G_i^∞ is a Gröbner basis for $<_U$
  - $W_\infty = \cup_{i=0}^{n-d} \boldsymbol{V}(G_i^\infty)$
- **Begin**
  - **Set** $G_i^\infty = G_{U,X} \cap \mathbb{Q}[U]$ for $i = d+1, ..., n$
  - **for** $g \in G_{U,X}$
    - **if** $\exists i \in [d+1...n]$ and $\exists k \in \mathbb{N}^\star$ such that $\mathrm{LM}_{<_X}(g) = X_i^k$ **then** $G_i^\infty = G_i^\infty \cup \{\mathrm{LC}_{<_X}(g)\}$
  - **return** $(G_i^\infty, d+1 = 1, ..., n)$
- **End**

5.6.3 Le calcul des points critiques et singuliers

Le calcul de O_c et de O_{sing} est un point très sensible de l'algorithme général puisque c'est à ce moment que les stratégies de calcul divergent en fonction des propriétés du système ou de la sortie minimale imposée par l'utilisateur.

La fonction qui suit se contente de calculer $\left(\langle\mathcal{I}\rangle + \langle \text{Jac}_X^{n-\delta}(\mathcal{E})\rangle\right) \cap \mathbb{Q}[U]$ et $\langle\mathcal{I}_{\Pi_U}\rangle + \langle \text{Jac}_U^{\delta}(\mathcal{I}_{\Pi_U})\rangle$, renvoyant un message en fonction des observations pouvant être faites sur le système. Pour ne citer que ce qui n'a pas déjà été pointé jusqu'ici, si $\dim(\langle\mathcal{I}\rangle + \langle \text{Jac}_X^{n-\delta}(\mathcal{E})\rangle) < \delta$, alors on peut toujours remplacer $\overline{O_c}$ par $\boldsymbol{V}(\dim(\langle\mathcal{I}\rangle + \langle \text{Jac}_X^{n-\delta}(\mathcal{E})\rangle) < \delta)$ si on désire obtenir au final une variété discriminante qui ne soit pas nécessairement stricte. Même remarque pour O_{sing} lorsque $\dim(\langle\mathcal{I}_{\Pi_U}\rangle + \langle \text{Jac}_U^{\delta}(\mathcal{I}_{\Pi_U})\rangle) < \delta$.

Algorithm CRITICAL

- **Input** : $\mathcal{E}, G, G_\Pi, \delta, U, X$ as in Algorithm PREPROCESSING
- **Output** : G_c, G_{sing} and *Property* such that
  - G_c and G_{sing} are reduced Gröbner bases for $<_U$
  - if *Property=Minimal*, then $W_D = \overline{O_\infty} \cup \boldsymbol{V}(\langle G_c\rangle) \cup \boldsymbol{V}(\langle G_{\text{sing}}\rangle) \cup \overline{O_\mathcal{F}}$ is the minimal discriminant variety of $\mathcal{C}$ w.r.t. Π_U.
  - if *Property=PartialLarge*, then $\overline{O_\infty} \cup \overline{O_{\text{sd}}} \cup \boldsymbol{V}(\langle G_c\rangle) \cup \boldsymbol{V}(\langle G_{\text{sing}}\rangle) \cup \overline{O_\mathcal{F}}$ is a discriminant variety of $\mathcal{C}$ w.r.t. Π_U.
  - if *Property=NeedRadical*, then $\overline{O_\infty} \cup \overline{O_{\text{sd}}} \cup \boldsymbol{V}(\langle G_c\rangle) \cup \boldsymbol{V}(\langle G_{\text{sing}}\rangle) \cup \overline{O_\mathcal{F}}$ is not a discriminant variety of $\mathcal{C}$ w.r.t. Π_U (if $\mathcal{I} = \langle G\rangle$, $\dim(\boldsymbol{V}((\mathcal{I} + \text{Jac}_X^{n-\delta}(\mathcal{I})) \cap \mathbb{Q}[U])) \geq \delta$).

- **Begin**
  - **Compute** G_{jac}, the reduced Gröbner basis of $(\langle G\rangle \cup \{\text{Jac}_X^{n-\delta}(\mathcal{E})\})$ w.r.t. $<_{U,X}$
  - **Deduce** $G_c = G_{\text{jac}} \cap \mathbb{Q}[U]$
    - if $\delta < d$ compute G_{sing}, the reduced Gröbner basis of $(\langle G_\Pi\rangle \cup \{\text{Jac}_U^{d-\delta}(G_\Pi)\})$ w.r.t. $<_U$
    - else set $G_{\text{sing}} = \{1\}$
  - **If** $\dim(\langle G_c\rangle) < \delta$ **then**
    - **if** $n - \delta = \sharp\mathcal{E}$ **then return**$(G_c, G_{\text{sing}},\ \textit{Minimal})$
    - **else return**$(G_c,\ G_{\text{sing}},\ \textit{PartialLarge})$
  - **else return**$(G_c, G_{\text{sing}}, \textit{NeedRadical})$
- **End**

5.6.4 Le cas des systèmes les plus courants

L'algorithme qui suit a pour but premier de calculer le plus rapidement possible la variété discriminante minimale pour n'importe quel système « bien formulé », mais il a également pour vocation à être le composant principal de l'algorithme général. En fait, il s'appuie sur la fonction Critical qui permet de détecter quelques informations importantes sur les idéaux définissant le système et profite que certains résultats de la section 5.5 s'appliquent à des systèmes qui ne sont pas forcément « bien formulés » :

Algorithm Core

- **Input** : $\mathcal{E}, \mathcal{F}, U, X$
- **Output** : $G, G_\Pi, \delta, k, G_{D,1}, ..., G_{D,k}$ and *Property* such that
  - G is a reduced Gröbner basis for $<_{U,X}$ where $<_U$ and $<_X$ are Degree Reverse Lexicographic orderings;
  - $(G_{D,i})_{i=1...k}$ and I_Π are Gröbner bases for $<_U$;
  - if *Property*=*Minimal*, then $\cup_{i=1}^{k} \boldsymbol{V}(\langle G_{D,i}\rangle)$ is the minimal discriminant variety of $\mathcal{C}$ w.r.t. Π_U;
  - if *Property*=*PartialLarge*, then $O_{\text{sd}} \cup_{i=1}^{k} \boldsymbol{V}(\langle G_{D,i}\rangle)$ is a discriminant variety of $\mathcal{C}$ w.r.t. Π_U, which has the same components of dimension δ, if any, as the minimal one;
  - if *Property*=*NeedRadical*, then $O_{\text{sd}} \cup O_c \cup O_{\text{sing}} \cup_{i=1}^{k} \boldsymbol{V}(\langle G_{D,i}\rangle)$ is the minimal discriminant variety of $\mathcal{C}$ w.r.t. Π_U.
- **Begin**
  - $\delta, G, G_\Pi, G_\mathcal{F}$=PrePocessing($\mathcal{E}, \mathcal{F}, U, X$)
  - **if** ($G_\Pi = \{1\}$) **then return**(G,G_Π,δ,1,G_Π,*Minimal*)
  - $(G_i^\infty)_{i=1...n-d}$=PropernessDefects(G,U,X)
  - **if** $G_i^\infty = G_\Pi$ for some i in $\{1, ..., n-d\}$ **then return**(G,G_Π,δ,1,G_Π,*Minimal*)
  - G_c, G_{sing},*Property*=Critical($\mathcal{E}, G, G_\Pi, \delta, U, X$)
  - *if Property*=*NeedRadical*, **then**
    return($G, G_\Pi, \delta, n-d+1, G_\mathcal{F}, (G_i^\infty)_{i=1...n-d}$,*Property*)
  - **else return**($G, G_\Pi, \delta, n-d+3, I_\mathcal{F}, (G_i^\infty)_{i=1...n-d}$,$G_c, G_{\text{sing}}$,*Property*)
- **End**

5.6.5 L'algorithme général dans le cas complexe

Une fois que l'on a l'algorithme Core, qui permet de détecter les systèmes bien formulés et

d'en calculer la variété discriminante minimale, mais également, dans certains autres cas, de calculer partiellement ou en totalité une variété discriminante (éventuellement non stricte), l'algorithme général consiste simplement à utiliser le plus parcimonieusement possible un algorithme de décomposition pour permettre de résoudre systématiquement le cas général en fonction de la requête de l'utilisateur.

Algorithm DISCRVAR

- **Input:**
  - $\mathcal{E}, \mathcal{F}, U, X$ as in the above algorithms
  - *Request*, which may be set to *Minimal* if a minimal discriminant variety is requested, *Large* if a discriminant variety (non necessarilly minimal) is requested and *Partial* if one allows to ignore (at least partially) O_{sd} in the final result.
  - DECOMPOSE : a function which computes and décomposes any ideal $\sqrt{\mathcal{I}} \subset \mathbb{Q}[U, X]$ as an intersection of equidimensional and radical ideals without commun associated primes.
- **Output:** $G_{D,1}, ..., G_{D,k}$, *Property* such that
  - $G_{D,i}, i = 1, ..., k$ are Gröbner bases for $<_{U,X}$
  - if *Property* = *Minimal*, $\bigcup_{i=1}^{k} \boldsymbol{V}(\langle G_{D,i} \rangle)$ is the the minimal discriminant variety of $\mathcal{C}$ w.r.t. Π_U.
  - if *Property* = *Large*, $\bigcup_{i=1}^{k} \boldsymbol{V}(\langle G_{D,i} \rangle)$ is a (non necessarily minimal) discriminant variety of $\mathcal{C}$ w.r.t. Π_U;
  - if *Property* = *PartialLarge*, $O_{\mathrm{sd}} \cup \bigcup_{i=1}^{k} \boldsymbol{V}(\langle G_{D,i} \rangle)$ is a discriminant variety of $\mathcal{C}$ w.r.t. Π_U.
  - *Property* may not take the same values as *Request*, indicating possibly that a stronger result than asked has been obtained (for example the user may ask for a non minimal discriminant variety but receive the minimal one).
- $G, G_\Pi, \delta, k, (G_{D,i})_{i=1,...,k}$, *Property*=CORE$(\mathcal{E}, U, X)$
- **if** *Property* = *Minimal* or *Request* = *Property* **then return** $((G_{D,i})_{i=1,...,k}$, *Property* $)$
- $(G_i)_{i=1,...,m}$ = DECOMPOSE(G)
- **if** *Property* = *PartialLarge* and *Request* = *Large* **then**
  - $G_D = \{G_{D,i}, i = 1, ..., k\}$
  - **for** $i = 1, ..., m$ **do if** $\dim(G_i) < \delta$ **then**
    - $G_\Pi^{(i)} = G_i \cap \mathbb{Q}[U]$
    - **if** $\dim(G_\Pi^{(i)}) = \dim(G_i)$ **then** $G_D = G_D \cup \{G_\Pi^{(i)}\}$
  - **return**$(I_D$, *Large*$)$
- $I_D = \{G_{D,i}, i = 1, ..., n - d + 1\}$ *(at this stage we have* $\boldsymbol{V}(G_D) = W_\infty \cup W_\mathcal{F}$*)*
- **for** $i = 1, ..., m$ **do if** $\dim(G_i) \leqslant \delta$ **then**
  - $G_\Pi^{(i)} = G_i \cap \mathbb{Q}[U]$
  - **if** $\dim(G_\Pi^{(i)}) = \dim(G_i) < \delta$ **then** $G_D = G_D \cup \{G_\Pi^{(i)}\}$
  - **if** $\dim(G_\Pi^{(i)}) = \dim(G_i) = \delta$ **then** $G_D = G_D \cup \{$ CRITICAL$(G_i, G_i, G_\Pi^{(i)}, \delta, U, X)\}$
- **for** $i = 1, ..., m$ **do for** $j = i + 1, ..., m$ **do**
  - **if** $\delta = \dim(G_i) = \dim(G_j) = \dim(G_\Pi^{(i)}) = \dim(G_\Pi^{(j)})$ **then**
    - $G_D = G_D \cup \{(G_i + G_j) \cap \mathbb{Q}[U]\}$
    - **if** $\delta < d$ **and** $G_\Pi^{(i)} \neq G_\Pi^{(j)}$ **then**
      - $(K_{i'})_{i'=1,...,m'}$ = DECOMPOSE $(G_\Pi^{(i)} + G_\Pi^{(j)})$
      - **for** $i' = 1, ..., m'$ **do if** $\dim(K_{i'}) < \delta$ **then** $G_D = G_D \cup \{K_{i'}\}$
- **if** DECOMPOSE provides a certified irredundant output
  - **then** *Property* = *Minimal* **else** *Property* = *Large*
- **return**$(G_D$, *Property*$)$

5.6.6 L'algorithme général dans le cas réel

Comme on l'a vu dans la section 5.3, il n'existe pas de variété discriminante minimale dans le cas réel, mais on peut déduire une variété discriminante réelle d'une variété discriminante (complexe) dès lors que $\left(\mathcal{C} \setminus \Pi_U^{-1}(\Pi_U(\mathcal{C}) \setminus W_D)\right) \cap \mathbb{R}^n$ est non vide. L'algorithme suivant effectue ce test et, le cas échéant, construit un un ensemble constructible $\mathcal{C}'$ contenant $\mathcal{S}$ mais ayant cette cette propriété (selon la stratégie décrite dans la section 5.3).

Algorithm RealDiscrVar

- **Input:**
  - $\mathcal{E}, \mathcal{F}, U, X$ as in the preceding algorithms.
  - *Request*, which may be set to *Minimal* if a minimal discriminant variety is requested, *Large* if a discriminant variety is requested and *PartialLarge* if one allows W_{sd} to be incomplete;
  - SamplePoints(Eq,Ineq,k) a function which computes at least one point in each connected component of the semialgebraic set of $\mathbb{R}^k$ $\{g=0, g \in \mathrm{Eq}, f>0, f \in \mathrm{Ineq}\}$.
- **Ouput:** $G_{D,1}, ..., G_{D,k}$, *Property* such that
  - $G_{D,i}, i=1, ..., k$ are Gröbner bases for $<_{U,X}$
  - if *Request* = *DiscrVar*, then $\bigcup_{i=1}^{k} \boldsymbol{V}(\langle G_{D,i}\rangle)$ is a discriminant variety of $\mathcal{S}$ w.r.t. Π_U.
  - if *Request* = *Large*, then $\bigcup_{i=1}^{k} \boldsymbol{V}(\langle G_{D,i}\rangle)$ is a (non necessarilly minimal) discriminant variety of $\mathcal{S}$ w.r.t. Π_U;
  - if *Request* = *PartialLarge*, $O_{\mathrm{sd}} \bigcup_{i=1}^{k} \boldsymbol{V}(\langle G_{D,i}\rangle)$ is a discriminant variety of $\mathcal{S}$ w.r.t. Π_U.
- $\delta, G, G_\Pi, G_\mathcal{F}$=PreProcessing($\mathcal{E}, \mathcal{F}, U, X$)
- **if** ($G_\Pi = \{1\}$) **then return**(G,G_Π,δ,1,G_Π,*DiscrVar*)
- $(G_i^\infty)_{i=1...n-d}$=PropernessDefects(G,U,X)
- **if** $G_i^\infty = G_\Pi$ for some i in $\{1, ..., n-d\}$ **then**
  - Compute $Y \subset [U, X]$ a maximal subset of transcendental variables for $I = \langle G\rangle$;
  - $G'_\Pi, G'_{D,1}, ..., G'_{D,k}$, *Property*=DiscrVar($G, G_\mathcal{F}, Y, [U, X] \setminus Y$, *Large*)
  - if SamplePoints($G'_\Pi \cup_{i=1}^{k} G'_{D,i}, \mathcal{F}, \#Y) \neq \emptyset$ then **return**(I_Π)
  - else **return**(RealDiscrVar($G \cup_{i=1}^{k} G_{D,i}, \mathcal{F}, U, X$))
- **else** return(DiscrVar($\mathcal{E}, \mathcal{F}, U, X$,*Large*))

Notons que la dernière ligne est un raccourci pour éviter une description trop longue : en pratique, il faut tenir compte du fait que $\delta, G, G_\mathcal{F}, G_\Pi$ et $G_i^\infty, i = 1...n-d$ ont déjà été calculés.

5.6.7 Une CAD optimisée pour l'utilisation des variétés discriminantes

La Décomposition Cylindrique Algébrique (CAD) est la seule possibilité connue permettant de donner une description exploitable en pratique des composantes connexes de $\Pi_U(\mathcal{S}) \setminus (W_D \cap \mathbb{R}^d)$. Rappelons que l'on parle toujours d'une CAD de $\mathbb{R}^n$ adaptée à une famille de polynômes $\mathcal{P}$ de $\mathbb{R}[Y_1, ..., Y_n]$ et que ce procédé calcule une décomposition de $\mathbb{R}^n$ en cellules (semi-algébriques homéomorphes à des pavés de tailles arbitraires $(0,1)^k$ $k = 0...n$) dans lesquels tout polynôme de $\mathcal{P}$ est de signe constant ($= 0, > 0, < 0$). Les cellules sont données sous la forme d'un point et de fonctions semi-algébriques.

Si l'on veut caractériser les cellules de $\Pi_U(\mathcal{S}) \setminus (W_D \cap \mathbb{R}^d)$ avec une CAD, une méthode naïve est la suivante :

- on calcule une CAD de $\mathbb{R}^d$ adaptée à l'ensemble des polynômes définissant $\overline{\Pi_U(\mathcal{C})}$ et W_D, c'est à dire, par exemple, ceux des diverses bases de Gröbner calculées par les algorithmes des sections précédentes. Appelons G_1, ..., G_l les bases de Gröbner obtenues telles que $W_D = \cup\, \boldsymbol{V}(\langle G_i\rangle)$ et G_Π la base de Gröbner telle que $\overline{\Pi_U(\mathcal{C})} = \boldsymbol{V}(\langle G_\Pi\rangle)$: on calcule alors une CAD adaptée aux polynômes de l'ensemble $\cup_{i=1...l} G_i \cup G_\Pi$.
- pour chaque cellule de la CAD, on filtre celles qui ne sont pas dans $\Pi_U(\mathcal{C} \cap \mathbb{R}^n) \setminus (W_D \cap \mathbb{R}^d)$: en supposant que u soit le point test de la cellule considérée, on la rejette si il existe au moins un ensemble G_i tel que $\forall p \in G_i, p(u) = 0$, ou alors si $\exists p \in G_\Pi$ tel que $p(u) \neq 0$.

- à ce stade, il ne reste plus que des cellules homéomorphes à $(0,1)^\delta$ dont l'union est exactement $\Pi_U(\mathcal{C}\cap\mathbb{R}^n)\setminus(W_D\cap\mathbb{R}^d)$.
- pour chaque cellule restante, on filtre celles qui sont dans $\mathcal{S}$: en supposant que u soit le point test de la cellule considérée, on résout le système $\mathcal{S}_{|U=u}$, par exemple en calculant une RUR étendue du système zéro-dimensionnel $\mathcal{E}_{|_{U=u}}$ et des polynômes $\mathcal{F}_{|U=u}$, et on rejette la cellule si il n'admet pas de solutions.
- au final, il ne reste plus que des cellules homéomorphes à $(0,1)^\delta$, dont l'union est exactement $\Pi_U(\mathcal{S})\setminus(W_D\cap\mathbb{R}^d)$ et on connaît même le nombre (constant) de solutions de $\mathcal{S}$ au dessus de ces cellules.

Une remarque importante, d'un point de vue pratique, est que la CAD calcule des cellules de toutes dimensions alors que l'on a besoin que des cellules de dimension $\delta=\dim(\overline{\Pi_U(\mathcal{C})})$. Nous reviendrons rapidement sur le cas général à la fin de cette section, l'objectif étant avant tout de traiter le cas des systèmes « bien formulés ».

En fait, l'unique hypothèse faite dans ce qui suit est de supposer que $\overline{\Pi_U(\mathcal{C})}=\mathbb{C}^d$ et force est de constater que le cas général n'a que peu d'intérêt, surtout si ces paramètres représentent des grandeurs physiques.

Dans le cas d'un système ou $\overline{\Pi_U(\mathcal{C})}=\mathbb{C}^d$ (donc en particulier dans le cas des systèmes bien « formulés »), les cellules utiles d'une CAD sont alors celles de plus grande dimension. Si on utilise l'algorithme de Collins [30] ou l'une de ses nombreuses variantes, la construction des cellules de dimension non maximale (calcul des points tests et des fonctions semi-algébriques) nécessite la résolution de polynômes en une variable dont les coefficients sont des nombres algébriques, alors que ce qui n'est pas le cas pour les cellules de dimension maximale ou tous les coefficients sont rationnels.

La CAD se décompose en 2 étapes : une phase de projection et une phase de remontée. A la k-ième étape de la phase de projection, on suppose que l'on a un ensemble $\mathcal{P}_k\subset\mathbb{Q}[U_k,...,U_d]$ (à l'étape 0, $\mathcal{P}_0$ est constitué des polynômes que l'on veut étudier). On note, pour tout $p,\ q\ \in\mathbb{Q}[U_k,\ ...,\ U_d]$, $\deg_{U_k}(p)$ le degré de p U_k, $\mathrm{lc}_{U_k}(p)$ son coefficient de tête relativement à U_k, et, $\mathrm{sr}_j(p,\ q)$ le j-ième coefficient sous-résultant principal de p et q. La $k+1$-ième étape de projection consiste à construire $\mathcal{P}_{k+1}=\mathrm{Proj}(\mathcal{P}_k)$, le plus petit ensemble de $\mathbb{Q}[U_{k+1},...,U_d]$ tel que :

CAD-Projection

- [A] pour $p\in\mathcal{P}_k$, $\deg_{U_k}(p)=d\geq 2$, $\mathrm{Proj}(\mathcal{P}_k)$ contient $\mathrm{sr}_j(p,\frac{\partial p}{\partial U_k})$ (non constants) pour $j=0,...,d$.
- [B] pour $p\in\mathcal{P}_k$, $q\in\mathcal{P}_k$, $\mathrm{Proj}(\mathcal{P}_k)$ contient $\mathrm{sr}_j(p,\ q)$ (non constants) pour $j=0,\ ...,\ \min(\deg_{U_k}(p),\deg_{U_k}(q))$.
- [C] pour $p\in\mathcal{P}_k$, tel que $\deg_{U_k}(p)\geq 1$ et que $\mathrm{lc}_{U_k}(p)$ soit non constant, $\mathrm{Proj}(\mathcal{P}_k)$ contient $\mathrm{lc}_{U_k}(p)$ et $\mathrm{Proj}(\mathcal{P}_k\setminus\{p\}\cup\{p-\mathrm{lc}_{U_k}(p)\})$.
- [D] pour $p\in\mathcal{P}_k$ tel que $\deg_{U_k}(p)=0$ et p non constant, $\mathrm{Proj}(\mathcal{P}_k)$, contient p.

L'astuce utilisée dans dans **FR**-[33] est de remarquer que les zéros de $p-\mathrm{lc}_{U_k}(p)$ se projettent tous dans O_∞ et qu'il est donc tout à fait inutile de considérer les polynômes $p-\mathrm{lc}_{U_k}(p)$ (étape [D]), les $\mathrm{sr}_j(p,\ q)$ (étape [B]) et $\mathrm{sr}_j(p,\frac{\partial p}{\partial U_k})$ (étape [A]) pour $j=1...d$ puisqu'ils ne servent qu'à décomposer des régions contenues dans la variété discriminante. L'étape de projection peut donc être modifié comme suit :

- [A'] pour $p\in\mathcal{P}_k$, $\deg_{U_k}(p)=d\geq 2$, $\mathrm{Proj}(\mathcal{P}_k)$ contient $\mathrm{Discriminant}(p,U_k)$
- [B'] pour $p\in\mathcal{P}_k$, $q\in\mathcal{P}_k$, $\mathrm{Proj}(\mathcal{P}_k)$ contient $\mathrm{Resultant}(p,q)$
- [C'] pour $p\in\mathcal{P}_k$, tel que $\deg_{U_k}(p)\geq 1$ et que $\mathrm{lc}_{U_k}(p)$ soit non constant, $\mathrm{Proj}(\mathcal{P}_k)$ contient $\mathrm{lc}_{U_k}(p)$
- [D'] pour $p\in\mathcal{P}_k$ tel que $\deg_{U_k}(p)=0$ et p non constant, $\mathrm{Proj}(\mathcal{P}_k)$, contient p.

La phase de remontée dans la CAD classique s'effectue elle aussi récursivement en partant avec $\mathcal{P}_k = \mathcal{P}_d$:

CAD-Remontée

- (1) on « calcule » les racines réelles de tous les polynômes de $\mathcal{P}_k$ et on les trie;
- (2) on prend un point dans chaque intervalle défini par (1);
- (3) on remplace U_k par (1) puis (2) dans $\mathcal{P}_{k-1}...\mathcal{P}_1$ et on « remonte » ensuite $\mathcal{P}_{k-1}$ qui est constitué de polynômes en une variable.

L'étape (1) n'a pour but que de caractériser des cellules de dimension non maximales (une des coordonnées est fixée) et peut donc être oubliée dans notre cas. Notons que c'est elle qui engendre des calculs nécessitant la résolution de polynômes en une variable à coefficients réels algébrique, ce qui est certainement le calcul le plus délicat en pratique de l'algorithme global. Cette phase de remontée peut donc être réduite, dans notre cas à :

- (2') on prend un point dans chaque intervalle entre 2 racines réelles de l'ensemble des polynômes de $\mathcal{P}_k$;
- (3') on remplace U_k par (2) dans $\mathcal{P}_{k-1}...\mathcal{P}_1$ et on « remonte » ensuite $\mathcal{P}_{k-1}$ qui est constitué de polynômes en une variable.

En termes d'efficacité, l'algorithme simplifié (calcul des d projections) reste certes doublement exponentiel en le nombre de variables (ici ce sont les paramètres du problème initial), mais on peut voir que la majorité des calculs sont évités dans la phase de projection et que les calculs les plus difficiles sont omis dans la phase de remontée.

Cette stratégie s'est en particulier révélée très efficaces pour décrire le complémentaire de variétés discriminantes pour des problèmes dépendant de 3 ou 4 paramètres [32], **FR**-[33]. Il faut noter qu'elle partage le même type d'ingrédients que des méthodes telles que [130] ou [28].

Lorsque $\overline{\Pi_U(\mathcal{C})} \subsetneqq \mathbb{C}^d$, les choses sont plus compliquées et plusieurs stratégies sont possibles :

- augmenter le nombre de paramètres de sorte à se ramener au cas ou $\overline{\Pi_U(\mathcal{C})} = \mathbb{C}^d$, mais selon la question posée, l'étude de $\mathcal{C}$ peut devenir délicate puisque les discussions fonction des valeurs des paramètres sont moins directes.
- simplement ignorer G_Π, mais intersecter avec G_Π les polynômes que l'on ajoute dans la phase de projection de la CAD : la gestion des intersections est complexe et peut-être très coûteuse en pratique, mais le gain sur le résultat final est facile à exploiter;

Il n'y a pas encore suffisamment d'applications ayant nécessité ce type de calcul pour permettre un choix clair de stratégie.

5.7 Conclusion et perspectives

Dans le cas complexe, la variété discriminante minimale d'un ensemble constructible est un objet « optimal » et a priori incontournable dès lors que l'on veut au moins étudier le nombre de solutions du problème en fonction de ses paramètres. Dans [93], un algorithme simplement exponentiel (temps de calcul et taille de l'objet calculé) permettant le calcul d'une variété discriminante minimale dans le cas de systèmes bien formulés est proposé, reprenant une stratégie similaire à celle que l'on peut trouver dans [63].

Dans le cas réel, la situation est moins claire, mais on peut semble-t-il se convaincre qu'il s'agit la encore du bon objet (section 5.3).

D'un point de vue algorithmique, le cas des systèmes biens formulés est le plus éprouvé. En combinant l'algorithme Core et une CAD épurée, il a été possible de résoudre des cas hors de portée des autres méthodes (voir chapitre 6). En pratique on peut observer que pour ce type de systèmes, c'est le calcul des valeurs critiques « au sens large » de la projection sur l'espace des paramètres qui est en général le plus difficile.

Même si l'on peut déjà proposer un algorithme pour le cas général, celui ci est très perfectible, surtout pour ce qui est de la décomposition des idéaux. En effet, pour l'heure, la solution privilégiée est une décomposition paresseuse basée sur un calcul d'ensemble triangulaires réguliers et séparables (permettant une décomposition en idéaux équi-dimensionnels et radicaux). Ceci dit, la connaissance de certaines composantes de la variété discriminante minimale, en particulier O_∞ devraient assurément permettent de faire évoluer les algorithmes classiques de décomposition primaire pour permettre le calcul de décompositions moins fines mais adaptées au problème.

Pour l'heure, une limitation à l'utilisation d'une variété discriminante pour l'étude d'un constructible ou d'un semi-algébrique est la description de son complémentaire (dans la clôture de Zariski de l'image de la projection de l'ensemble étudié). En effet seule la CAD permet, en pratique, de décrire sous une forme exploitable cet ensemble. Malgré l'adaptation proposée dans ce chapitre et malgré l'élimination des indéterminées en une seule passe (calcul de variété discriminante), une augmentation sensible du nombre de paramètres (disons plus de 5) aura rapidement raison de l'utilisation de la CAD, même modifiée, simplement à cause du nombre de cellules calculées.

Chapitre 6
Quelques applications

6.1 Systèmes zéro-dimensionnels

Dans les deux applications proposées pour illustrer le travail réalisé sur les systèmes zéro-dimensionnels, la première est historique (modèle géométrique direct des robots parallèles) puisqu'elle sert depuis très longtemps d'étalon pour les méthodes de calcul exact. La deuxième est la plus récente (étude de ridges en géométrie algorithmique) et la solution proposée prend le contre pied de l'état de l'art (essentiellement calcul différentiel et méthodes numériques).

6.1.1 Le problème géométrique direct des robots parallèles

Pour faire simple, un robot parallèle (voir [91]), que nous supposerons toujours être un hexapode, est constitué de deux ensembles rigides, la plate-forme et la base sur lesquels sont positionnées 6 rotules sphériques. Un exemple typique est la plate-forme de Gough [62].

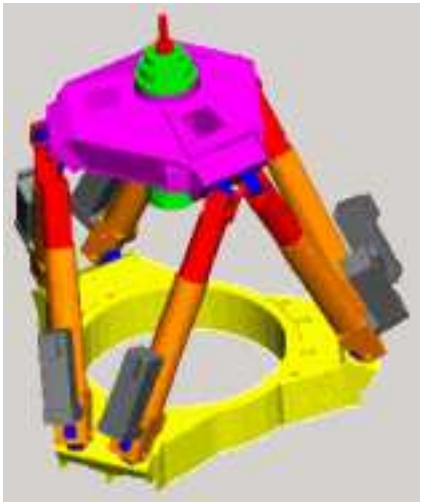


Figure 6.1. Un exemple d'hexapode

La plate-forme (resp. la base) est donnée par les coordonnées des 6 points $M_i, i = 1...6$ (resp. A_i, i=1..6) que l'on peut assimiler aux centres des rotules qui y sont présentes dans un repère $\mathcal{R}_p$ (resp. $\mathcal{R}_b$) qui lui est liée : les positions relatives des M_i (resp. A_i) sont donc constantes. La base et la plate-forme sont reliées par 6 jambes motorisées que l'on peut assimiler aux segments (A_i, B_i), dont les longueurs $L_i = \|A_iB_i\|$ peuvent varier. En supposant que la base ait une position fixe, bouger la plate-forme revient alors à faire varier les longueurs de jambes. Ce type de robot peut être utilisé pour diverses tâches, comme par exemple :

- les simulateurs de vol : on pose une maquette de cabine de pilotage sur la nacelle, la variation des longueurs des jambes induira alors des mouvements divers de cette cabine;

- les antennes de réception satellite peuvent être posées également sur la plate-forme d'un tel mécanisme qui en permettra alors l'orientation;

- les machine-outils : on peut fixer un outil d'usinage (polissage, perçage) sur la nacelle et ainsi le déplacer;

- ..

De nombreux problèmes algébriques sont rattachés à l'étude de ce type de mécanismes, liés à la conception, la calibration, la dynamique du système, mais nous n'aborderons, dans cette section, qu'à l'un des problèmes liés à l'étude purement cinématique.

Il y a deux problèmes cinématiques duaux sous-jacents à l'étude de tels robots :

- le problème géométrique inverse : étant données la position de la base dans un repère fixe (c'est à dire les coordonnées des A_i) ainsi que la position de la nacelle dans ce même repère, calculer la longueur des segments (A_i, B_i).

- le problème géométrique direct : étant donnés la géométrie de la base , c'est à dire les coordonnées des A_i dans un repère lié à la base), la géométrie de la plate-forme, c'est à dire les coordonnées des M_i dans un repère lié à la plate-forme ainsi que qu'un jeu de longueurs de jambes $L_i = \|A_i M_i\|$, déterminer la position de la plate-forme, c'est à dire les coordonnées des points M_i, dans un repère fixe.

Le problème géométrique inverse est typiquement l'un des problèmes qu'il faudra résoudre pour commander le robot : on se donne une trajectoire pour la nacelle (de sorte, par exemple, à ce qu'un outil décrive une courbe de l'espace avec une orientation fixée), c'est à dire un ensemble de 6 fonctions $M_i(t)$ définissant la position des points M_i dans un repère fixe en fonction du temps, et l'on déduit les fonctions d'élongation des jambes $L_i(t)$ correspondantes. Les fonctions $L_i(t)$ seront transformées en consignes pour les moteurs permettant de modifier les longueurs des 6 jambes. Si on ignore les problèmes possibles de collisions de jambes, le problème géométrique inverse se résout simplement, puisqu'une formule simple permet de connaître les $L_i(t)$ en fonction des A_i et $M_i(t)$.

Le problème géométrique direct est connu pour être délicat, puisqu'il peut admettre plusieurs solutions. En effet, si on suppose fixés un jeu de longueurs de jambes et des géométries pour la plate-forme et la base, il peut y avoir, sauf pour un robot dégénéré, jusqu'à 40 positions possibles pour des valeurs « génériques » des longueurs de jambes ([81],[95],[106],...). Il faut noter que pour certaines configurations spécifiques, le système peut admettre une infinité de postures, ce qui permet, par exemple, de faire décrire une courbe au robot sans actionner les moteurs : nous tiendrons compte de ce cas possible, mais il ne fait pas partie de l'étude.

Une conséquence est que, pour des fonctions d'élongation $L_i(t)$ données, un point de la nacelle (par exemple le point d'ancrage d'un outil) peut décrire 40 courbes distinctes, celles-ci pouvant se croiser ou simplement être singulières. La conséquence de la présence d'une singularité ou d'un croisement est que le comportement du robot passant en un tel point est structurellement imprévisible : il n'y a aucune raison pour qu'il ne change pas de trajectoire ou de branche de trajectoire.

Si on suppose que les $L_i(t)$ sont des fonctions algébriques dépendant du temps, les trajectoires possibles pour un point de la nacelle le sont également, si bien qu'en théorie, il s'agit simplement d'étudier ces courbes algébriques. En pratique, nous sommes loin de pouvoir calculer la topologie de ce type de courbes pour la simple raison que nous ne savons pas les calculer exactement en fonction de t. Notons d'ailleurs qu'un calcul de ce type est peut-être définitivement à proscrire compte tenu de la taille prévisible des objets à manipuler (degré en t). Par contre, quelques méthodes basées sur l'analyse par intervalles permettent de garantir, à la conception du robot, l'absence a priori de telles singularités dans un espace de travail ou encore de suivre une trajectoire de façon certifiée tant qu'elle est régulière.

La vocation première de la résolution du modèle géométrique direct, par une méthode ayant les spécifications de celle développée au chapitre 3, est donc plutôt de fournir un outil de diagnostique ponctuel (par exemple confirmer ou infirmer la présence de deux solutions très proches), supposant que les L_i sont des scalaires et non des fonctions.

Notation 6.1. *Pour des points O et C arbitrairement choisis sur la base et la plate-forme, on note :*

- *R_f: un repère cartésien de centre O lié à la base.*
- *R_m: un repère cartésien de centre C lié à la plate-forme.*
- *A_i,i=1..6: les points de la base ou sont situés les joints (assimilés à des points)*
- *B_i,i=1..6: les points de la plate-forme ou sont situés les joints (assimilés à des points).*
- *L_i,i=1..6 les distances $\|B_iA_i\|$.*

Pour un vecteur $\vec{v} \in \mathbb{R}^3$ (resp. un point $M \in \mathbb{R}^3$), on note $\vec{v}_{|R}$ (resp. $M_{|R}$) ses coordonnées dans le repère cartésien R.

Pour tout point $N \in \mathbb{R}^3$, on note NA *(resp.* NB*) la matrice dont les colonnes sont les vecteurs $\overrightarrow{\mathrm{NA}_i}$ (resp. $\overrightarrow{\mathrm{NB}}_i$)*

Un hexapode est alors entièrement décrit par

- la géométrie de sa base, c'est à dire la matrice $\mathrm{OA}_{|R_f}$;
- la géométrie de sa plate-forme, c'est à dire la matrice $\mathrm{CB}_{|R_m}$;

Le problème géométrique direct peut alors s'exprimer le la manière suivante : étant donnés $\mathrm{OA}_{|R_f}$, $\mathrm{CB}_{|R_m}$ et $L_i, i = 1...6$, trouver $\mathrm{OB}_{|R_f}$.

Il existe de nombreuses modélisations de ce problème et les plus adaptées au calcul algébrique se regroupent en deux catégories :

- **équations sur les positions.** La position dans l'espace d'un corps rigide tel que la base ou la plate-forme d'un hexapode est complètement définie par 3 ses points (distincts). Pour la plate-forme, on peut considérer par exemple B_1, B_2 et B_3 en supposant qu'ils soient distincts. Comme le corps est rigide, les coordonnées des points B_4, B_5, B_6 et C peuvent alors être déduites de celles de B_1, B_2 et B_3 dont les coordonnées définissent les 9 variables du système dont les équations modélisent simplement la préservation de la géométrie de la plate-forme et les contraintes sur les longueurs des jambes. La formulation de ce type la plus adaptée aux outils utilisés dans ce document est certainement celle de [80].
- **équations basées sur un déplacement**. Si il existe une position $\mathrm{OB}_{|R_f}$ telle que les contraintes $L_i^2 = ||A_iB_i||^2, i = 1...6$ soient vérifiées, alors il existe une rotation $\mathcal{R}$ telle que $\mathrm{OB}_{|R_f} = \mathrm{OC}_{|R_f} + \mathcal{R} \cdot \mathrm{CB}_{|R_m}$. Les entrées de la matrice $\mathcal{R}$ ainsi que les coordonnées de $\mathrm{OC}_{|R_f}$ seront les inconnues du système.

Le problème géométrique direct pour un hexapode sert de mesure de performance des algorithmes de résolution de systèmes algébriques depuis longtemps. Pour ne citer que les réalisations mettant en jeu des composants utilisés dans ce document, les bases de Gröbner ont été utilisées dans [48] pour établir une classification des mécanismes en fonction de leur géométrie (voir [70] ou [140] par exemple pour d'autres utilisation de méthodes symboliques, [92] pour l'utilisation d'analyse par intervalles, [132], [141] pour l'utilisation de méthodes d'homotopie). Il faut noter que la mise en équations est de type « équations sur les positions » et est optimisée pour éviter quelques étapes inutiles dans le calcul de base de Gröbner [80].

Il semble que la première stratégie ayant permis au moins de compter, par un algorithme exact, le nombre de solutions réelles du problème géométrique direct pour un robot général, soit celui proposé dans proposé dans **FR**-[112]. L'algorithme utilisé utilise la table de multiplication décrite dans le chapitre 3 pour construire la forme quadratique de Hermite q_1 et la réduire. Il a permit d'exhiber **FR**-[112],**FR**-[113] un robot de caractéristiques générales ayant 24 racines réelles, ce qui représentait le maximum connu en 1995, l'unique autre exemple avec autant de solutions réelles était un robot de base et de plate-forme planes [71]. Comme le montrent les résultats expérimentaux de **FR**-[112], les temps de calcul étaient prohibitifs (heures voir jours de calculs) et tous les robots ne pouvaient pas être être étudiés en moins de 24 heures de calcul).

Avec la mise au point de l'algorithme général calculant une représentation univariée rationnelle, les résultats ont été étendus puisque l'on pouvait alors isoler les solutions réelles et non plus simplement les compter. Les divers progrès réalisés sur les calculs de bases de Gröbner [43] ainsi que sur les calculs de RUR permettaient alors de résoudre la plupart des modèles géométriques directs en quelques minutes/heures.

L'arrivée de l'algorithme F_4 [44] pour les calculs de bases de Gröbner ainsi que l'injection de calcul multi-modulaire dans le cas d'idéaux *shape position* (ce qui est le cas standard pour ce type de problème) ont été un tournant pour ce problème puisque le calcul de RUR, base de Gröbner comprise, est alors descendu à quelques secondes. Ces progrès ont essentiellement montré que certains algorithmes étaient alors devenus alors les étapes bloquantes : l'isolation des zéros de polynômes en une variable par une stratégie basée sur un calcul de suites de Sturm (voir chapitre 2) ou le comptage des zéros réels du système via la construction et la réduction de la forme quadratique de Hermite (voir chapitre 3 - Théorème 3.10), malgré l'utilisation d'une table de multiplication optimisée.

La résolution du modèle géométrique direct pour les hexapodes n'est donc devenu un problème « facile » qu'avec l'arrivée de l'algorithme d'isolation présenté au chapitre 2, n'importe quel robot pouvant alors être « résolu » en quelques secondes.

Les derniers progrès **FR**-[49] ont porté sur la modélisation elle-même, en travaillant sur un modèle basé sur les déplacements mais surtout tirant profit des algorithmes de calcul de bases de Gröbner modernes [44], [50].

Une manière standard de modéliser un déplacement est d'utiliser les quaternions pour représenter la rotation $\mathcal{R}$ ([69],[111],[42], etc.). Pour paramétrer le groupe des rotations, on peut utiliser la transformation de Cayley; Si H est une matrice anti-symétrique :

$$H = \begin{bmatrix} 0 & a & b \\ -a & 0 & c \\ -b & -c & 0 \end{bmatrix}$$

Supposant que H n'ait pas 1 comme valeur propre,

$$\mathcal{R} = \frac{I+H}{I-H} = \begin{bmatrix} -\dfrac{-1-c^2+a^2+b^2}{1+c^2+a^2+b^2} & 2\,\dfrac{a-\mathrm{bc}}{1+c^2+a^2+b^2} & 2\,\dfrac{\mathrm{ac}+b}{1+c^2+a^2+b^2} \\ -2\,\dfrac{a+\mathrm{bc}}{1+c^2+a^2+b^2} & -\dfrac{a^2-1-b^2+c^2}{1+c^2+a^2+b^2} & -2\,\dfrac{-c+\mathrm{ab}}{1+c^2+a^2+b^2} \\ 2\,\dfrac{\mathrm{ac}-b}{1+c^2+a^2+b^2} & -2\,\dfrac{c+\mathrm{ab}}{1+c^2+a^2+b^2} & \dfrac{-b^2-c^2+1+a^2}{1+c^2+a^2+b^2} \end{bmatrix} \tag{6.1}$$

est une rotation. Inversement, si $\mathcal{R}$ est une rotation, alors $H = \frac{\mathcal{R}-I}{\mathcal{R}+I}$ est anti-symétrique, à condition toutefois que -1 ne soit pas une valeur propre de $\mathcal{R}$.

La relation (6.1), après suppression des dénominateurs, donne un système d'équations dépendant des 6 variables $[X, Y, Z, a, b, c]$. En choisissant les points C et O tels que $\mathrm{CB}_1 = 0$ et OA_1, on obtient alors :

$$\begin{aligned} L_1^2 &= ||\overrightarrow{\mathrm{OC}}_{|_{R_f}}||^2 \\ L_i^2 - L_1^2 &= ||\mathcal{R}\cdot\overrightarrow{\mathrm{CB}}_{i|_{R_m}} - \overrightarrow{\mathrm{OA}}_{i|_{R_f}}||^2 \;+2\; \left\langle \mathcal{R}\cdot\overrightarrow{\mathrm{CB}}_{i|_{R_m}} - \overrightarrow{\mathrm{OA}}_{i|_{R_f}}, \overrightarrow{\mathrm{OC}}_{|_{R_f}} \right\rangle, i = 2, \ldots 6 \end{aligned} \tag{6.2}$$

On peut remarquer que si on suppose $\mathcal{R}$ connu, $[X,\ Y,\ Z] = \overrightarrow{\mathrm{OC}}_{|_{R_f}}$ est solution d'un système linéaire à 5 équations et 3 inconnues). En particulier, il est donc intéressant de calculer une base de Gröbner du système pour un ordre d'élimination tel que $[X, Y, Z] > [a, b, c]$ puis de ne retenir que les polynômes dépendant de $[a,\ b,\ c]$, c'est à dire, exactement calculer une base de Gröbner de $\langle \mathcal{S} \rangle \cap \mathbb{Q}[a, b, c]$ si $\mathcal{S}$ est le système global à résoudre.

Les algorithmes F_4 et F_5 se prêtent particulièrement bien à la détection précoce des relations linéaires, si bien que ce calcul d'élimination s'avère peu coûteux en pratique **FR**-[49]. Si -1 n'est pas valeur propre de $\mathcal{R}$, la résolution du système initial peut donc se décomposer en 3 étapes :

- [A] résolution de G de $\langle \mathcal{S} \rangle \cap \mathbb{Q}[a, b, c]$
- [B] pour chaque solution de [A] résolution du système (6.2) linéaire en $[X,\ Y,\ Z]$ (après substitution des variables a, b, c).

Cette stratégie n'est correcte que si -1 n'est pas une valeur propre de $\mathcal{R}$. Dans le cas contraire, il pourrait manquer des solutions, ce qui ne peut arriver que si le nombre de zéros complexes de G, comptés avec multiplicités, est strictement inférieur à 40.

Dans le cas contraire, on utilise la version projective du système obtenu en homogénéisant les équations issues de (6.1) et (6.2) dont on cherche alors les « solutions à l'infini ». Si on note d la variable d'homogénéisation, calculer les solutions manquantes revient alors à effectuer le changement de variables $[a \longrightarrow \frac{a}{d},\ b \longrightarrow \frac{b}{d},\ c \longrightarrow \frac{c}{d}]$, puis à poser $d = 0$ dans l'expression (6.1), ce qui donne $\mathcal{R}_\infty = \begin{bmatrix} \frac{c^2-a^2-b^2}{c^2+a^2+b^2} & -2\,\frac{\mathrm{bc}}{c^2+a^2+b^2} & 2\,\frac{\mathrm{ac}}{c^2+a^2+b^2} \\ -2\,\frac{\mathrm{bc}}{c^2+a^2+b^2} & -\frac{a^2-b^2+c^2}{c^2+a^2+b^2} & -2\,\frac{\mathrm{ab}}{c^2+a^2+b^2} \\ 2\,\frac{\mathrm{ac}}{c^2+a^2+b^2} & -2\,\frac{\mathrm{ab}}{c^2+a^2+b^2} & -\frac{b^2+c^2-a^2}{c^2+a^2+b^2} \end{bmatrix}$ et enfin à combiner cette expression avec (6.2) pour obtenir un système, noté $\mathcal{S}_\infty = 0$ dans la suite, dont on cherchera alors les solutions telles que $a^2 + b^2 + c^2 \neq 0$.

En pratique, beaucoup d'optimisations sont possibles. La première remarque d'importance est que, sauf cas particulier, le système d'équations $\mathcal{S} = 0$ possède 40 solutions distinctes et l'idéal $\langle \mathcal{S} \rangle$ est *shape position* pour toutes les variables $[X,\ Y,\ Z,\ a,\ b,\ c]$. Ceci implique qu'en général on n'aura pas besoin d'étudier le cas dégénéré (calcul de $\mathcal{R}_\infty$), mais surtout que l'on pourra appliquer des stratégies multi-modulaires ou p-adiques pour le calcul d'une RUR de $\langle \mathcal{S} \rangle$ (voir Chapitre 3).

Une deuxième remarque est que le système $\mathcal{S}_\infty = 0$ est beaucoup plus facile à résoudre que $\mathcal{S} = 0$.

Du fait de la séparation de l'étude des solutions du système en deux étapes : résolution de $\langle \mathcal{S} \rangle \cap \mathbb{Q}[a, b, c]$ (et le cas échéant de $\langle \mathcal{S}_\infty \rangle \cap \mathbb{Q}[a, b, c]$) puis résolution du système linéaire en $[X,\ Y, Z]$, l'algorithme global est plus compliqué que la l'application directe de la méthode proposée dans le chapitre 3. Supposant $\langle \mathcal{S} \rangle \cap \mathbb{Q}[a,\ b,\ c]$ résolu, c'est à dire connaissant des produits d'intervalles isolant les solutions réelles, la suite des calculs d'effectue ainsi :

- [A] si le nombre de solutions complexes comptées avec multiplicités de $\langle \mathcal{S} \rangle \cap \mathbb{Q}[a, b, c]$ est < 40, on calcule les solutions de $\langle \mathcal{S}_\infty \rangle \cap \mathbb{Q}[a,\ b,\ c]$ et on rajoute les pavés isolants correspondants à ceux obtenus pour $\langle \mathcal{S} \rangle \cap \mathbb{Q}[a,\ b,\ c]$. Si ce nombre est infini, on est dans le cas d'un robot singulier (infinité de solutions au problème géométrique direct).
- [B] pour chaque solution, on substitue les variables [a,b,c] par les intervalles du pavé isolant correspondant dans $\mathcal{S} = 0$ et on vérifie qu'au moins un des mineurs 3×3 de ce système linéaire, calculé en utilisant une arithmétique par intervalles, est représenté par un intervalle ne contenant pas 0. Le cas échéant, on résout le système linéaire associé pour obtenir des intervalles contenant les valeurs des coordonnées [X,Y,Z] manquantes.

- [C] si tous les intervalles contenant les mineurs 3×3 (notés $M_i \in \mathbb{Q}(a, b, c)$ dans ce qui suit) contiennent 0, on résout symboliquement les systèmes $\mathcal{S} = 0 \cup M_i = 0$. Si il en existe au moins un pour lequel le système $\mathcal{S} = 0 \cup M_i = 0$ n'admet pas de solutions réelles, on raffine les pavés d'isolation des zéros de $\langle \mathcal{S} \rangle \cap \mathbb{Q}[a, b, c]$, on augmente la précision de l'arithmétique d'intervalles et on retourne à l'étape [A]. Pour une largeur suffisamment petite et une précision de l'arithmétique suffisamment grande, l'intervalle représentant le mineur non nul ne contiendra pas 0.
- [D] si tous les systèmes $\mathcal{S} = 0 \cup M_i = 0$ admettent au moins une solution réelle, le robot est singulier (infinité de solutions).

Les dernières expériences menées **FR**-[49] montrent une progression d'un facteur environ 1000 entre la version initiale **FR**-[112] et la version la plus récente **FR**-[49] de l'algorithme d'isolation des solutions pour le problème géométrique direct des hexapodes. Notons que désormais, un robot quelconque nécessite 1 à 2 secondes pour être résolu si la précision demandée reste raisonnable (coordonnées des solutions à 2^{-64} près).

Dans le résumée d'expériences suivant, on utilise les exemples :

- Le Robot de Dietmaier (le premier a 40 solutions réelles), obtenu par un procédé d'optimisation [38] :

```
Base:= [
  [0,0,0],
  [0.542805,0,0],
  [0.956919,-0.528915,0],
  [0.665885,-0.353482,1.402538],
  [0.478359,1.158742,0.107672],
  [-0.137087,-0.235121,0.353913]
]:
Platform:=[
  [0,0,0],
  [1.107915,0,0],
  [0.549094,0.756063,0],
  [0.735077,-0.223935,0.525991],
  [0.514188,-0.526063,-0.368418],
  [0.590473,0.094733,-0.205018]
]
```

  Les 40 solution sont obtenues pour les longueurs de jambes suivantes :
  `Legs:=[1,0.645275,1.086284,1.503439,1.281933,0.771071]:`
  Tous les nombres flottants seront convertis en rationnels, plus précisément on se donnera un entier δ et une fonction de conversion $f_\delta(x) = \frac{\lfloor 2^\delta x \rfloor}{2^\delta}$.

  Ainsi, dans ce qui suit, `dietmaier_` β est le système $\{\mathcal{S}_{\text{quat}}, \mathcal{S}_{\text{3pt}}, \mathcal{S}_{\text{rot}}\}$ convertit pour $\delta = \beta$. On peut voir que δ est trop petit, le système n'a pas 40 racines réelles: par exemple, lorsque $\delta = 17$ il n'y a que 38 solutions réelles.

- `Left Hand`. C'est une instance du robot « main gauche » de Jean-Pierre Merlet :

```
Base:=[
  [-9.704000000, 9.107000000, 0],
  [9.708000000, 9.110000000, 0],
  [12.75400000, 3.892000000, 0],
  [599/200, -12993/1000, 3/500],
  [-1503/500, -13, 1/200],
  [-12.77000000, 3.901000000, 1/1000]
]:
Platform:=[
  [-3003/1000, 7.290000000, 0],
  [601/200, 7.296000000, 0],
  [7.814000000, -1.053000000, 0],
  [4.812000000, -6.246000000, 3/1000],
  [-4.823000000, -6.257000000, 1/125],
  [-7.818000000, -1.057000000, 1/100]
]:
```

Pour les tests, nous prenons les longueurs de jambes :
`Legs:=[9999/500,4001/200,20007/1000,20007/1000,19993/1000,5001/250]:`

Cette configuration est intéressante puisqu'il s'agit d'un *cas exceptionnel* : le système a seulement 36 solutions complexes et, en particulier, 1 est toujours une valeur propre de $\mathcal{R}$:

$$\mathcal{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & ? & ? \\ 0 & ? & ? \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \mathcal{R}_2 \end{bmatrix}$$

$\mathcal{R}_2$ étant une rotation. Le calcul montre que l'ensemble des solutions se décompose en deux sous-ensembles :

i. l'un calculé avec $\mathcal{R}_\infty$: -1 est valeur propre de $\mathcal{R}$ (or $\mathcal{R}_2$) de multiplicité 2 pour 6 solutions complexes.

ii. l'autre avec $\mathcal{R}$: -1 n'est pas une valeur propre de $\mathcal{R}$ pour les 30 autres solutions complexes.

- `Left Hand` $+\varepsilon$ est une version perturbée du robot précédent : il a alors 40 solutions mais on remarquera que le nombre de solutions réelles est le même (8).
- `Spat 66` est un robot « générique » avec 40 solutions complexes (les coordonnées du robot ainsi que les longueurs des jambes sont choisies au hasard dans l'intervalle $[-1000, 1000]$). C'est ce type de robot qui est utilisé dans [48] et **FR**-[112]. Le calcul effectué dans **FR**-[112], que l'on peut considérer comme variante préliminaire des algorithmes proposés dans le chapitre 3 prenait plusieurs heures, voir jours. Cette ancienne stratégie a été exécutée sur des machines récentes de sorte à mesurer l'apport des améliorations algorithmiques : on observe des temps de calcul de l'ordre de 800 secondes :
  - Une base de Gröbner pour l'ordre DRL utilisant une variante optimisée de l'algorithme de Burchberger : 415 sec;
  - Un calcul de RUR utilisant des fonctions multi-symétriques comme dans **FR**-[113] : 243 sec;
  - Un calcul de suite de Sturm : 161 sec.

  Ceci est à comparer avec les 0.75 sec que l'on peut observer avec des algorithmes plus récents : F4 [44] pour les bases de Gröbner + utilisation de calcul multi-modulaire pour calculer la RUR (Chapitre 3)+ utilisation de la méthode de Descartes (Chapitre 2). Au bilan, on gagne un facteur environ 1000 sur une même machine.

Exemple	GB+RUR	ISO(32)	ISO(64)	Commentaire
`dietmaier_16`	1.0	0.20	0.28	40 complex roots / 38 real roots
`dietmaier_18`	1.1	0.21	0.32	40 real roots
`dietmaier_24`	1.4	0.25	0.36	"
`dietmaier_30`	1.8	0.28	0.40	"
`dietmaier_40`	2.0	0.31	0.43	"
`dietmaier_60`	2.2	0.32	0.46	"
`dietmaier_120`	2.9	0.40	0.56	"
`Spat66_quat`	0.7	0.00	0.01	40 complex roots / 0 real roots
`LeftHand` $\mathcal{R}$	0.4	0.12	0.16	30 complex roots / 8 real roots
`LeftHand` $\mathcal{R}_\infty$	0.04	0.00	0.01	6 complex roots / 0 real roots
`LeftHand` $+\varepsilon$	0.9	0.10	0.12	40 complex roots / 8 real roots

La colonne ISO(32) (resp. ISO(64)) indique le temps de calcul (en secondes) pour isoler les zéros réels du système à partir d'une RUR avec une précision (largeur des intervalles - voir Chapitre 3) inférieure à $\frac{1}{2^{32}}$ (resp. $\frac{1}{2^{64}}$), alors que la colonne GB+RUR donne le temps de calcul combiné pour une base de Gröbner puis une RUR.

6.1.2 Calcul de la topologie de ridges

Étant donnée une surface lisse, un ridge bleu (rouge) est une courbe le long de laquelle la courbure principale maximum (minimum) a un extremum en suivant sa ligne de courbure. Les ridges sont des lignes d'extrêmes de courbure et codent des informations importantes utilisées en segmentation, recalage, comparaison et analyse de surfaces, etc...

Étudier les ridges d'une surface nécessite l'évaluation de dérivées d'ordre trois et quatre, induisant des difficultés dès lors que l'on utilise du calcul approché. De plus, les ridges s'auto-intersectent et interagissent de façon complexe avec certains points de la surface (les ombilics) de la surface induisant des difficultés de nature topologique.

Un ridge est une courbe constituée des points ou l'une des courbures principales admet un extremum sur sa ligne de courbure. Sauf point particulier ou le ridge n'est pas défini (le point est alors nommé ombilic), il y a deux courbures principales à considérer (sinon les 2 courbures sont égales). Si on désigne par $k1$ et k_2 les courbures principales, en prenant comme convention que $k_1 \geq k_2$, un ridge est bleu (resp. rouge) si k_1 (resp. k_2) admet un extremum. Deux ridges de couleurs différentes s'intersectent à un point pourpre et un ridge est dit elliptique si il correspond à un maximum de k_1 ou un minimum de k_2, il est dit hyperbolique dans tous les autres cas (voir [26] pour une collection de contributions sur le sujet).

L'exemple ci-dessous illustre ces quelques définitions avec quelques conventions supplémentaires : les ridges verts correspondent à des minima de k_1, les jaunes à des maximal de k_2.

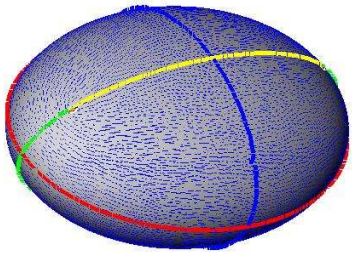


Figure 6.2.

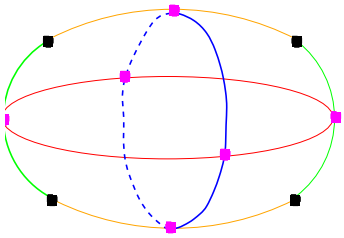


Figure 6.3.

La caractérisation des ridges et de leur couleur s'effectue a priori essentiellement de façon différentielle. En dehors des ombilics ou ils ne sont pas définis, les ridges associés à k_1 (resp. k_2) sont usuellement caractérisés par une équation du type $b_0 = 0$, (resp. $b_3 = 0$), b_0 (resp. b_3) étant la dérivée de la courbure principale k_1 (resp. b_3) par rapport à la direction de sa ligne de courbure.

Numériquement, le calcul des ridges, et leur caractérisation, se fait en recherchant les changements de signes des fonctions b_0 et b_3 sur une courbe, ce qui est donc intrinsèquement un problème délicat d'un point de vue calculatoire si on utilise du calcul approché. Les ridges ne sont définis que par morceaux (en dehors des ombilics), ce qui complique sévèrement le calcul. Outre les soucis de stabilité liés au calcul numérique, étudier le signe de b_0 ou de b_3 sur une portion de courbe impose de trouver une orientation cohérente, au moins pour les points extrêmes.

Bien que les résultats présentés dans cette section s'appliquent à des cas plus généraux, on s'intéresse aux surfaces paramétrées de la forme Φ: $(u, v, z) \longrightarrow (u, v, \phi(u, v))$ dans le pavé $[0, 1] \times [0, 1]$, ϕ étant un polynôme de $\mathbb{Q}[u, v]$ et les « challenges » proposés sont raisonnables puisque l'on se donne pour but d'étudier des polynômes ϕ de degré 3 à 5 en chacune des variables, c'est à dire les premiers cas intéressants dans un souci applicatif. L'exemple utilisé dans ce chapitre pour illustrer les résultats est :

- $\phi = 116\,u^4v^4\text{-}200\,u^4v^3{+}108\,u^4v^2\text{-}24\,u^4\ v\text{-}312\,u^3v^4{+}592\,u^3v^3\text{-}360\,u^3v^2{+}80\,u\ ^3v{+}252\,u^2v^4\text{-}504\,u^2v^3{+}324\,u^2v^2\text{-}72\,u^2v\text{-}56\,uv^4{+}112\,uv^3\text{-}72\,uv^2{+}16\,uv.$

Ceci nous donne pour Φ :

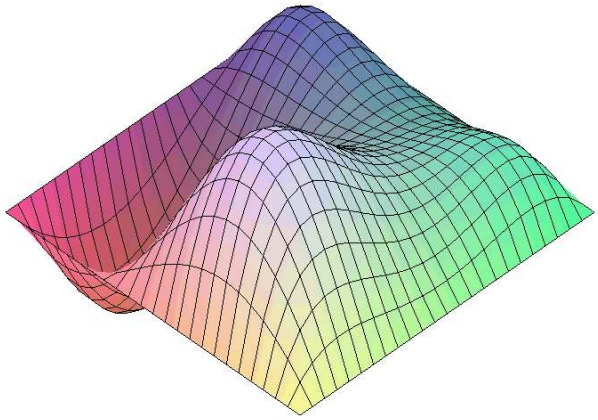


Figure 6.4.

Dans le cas « générique », les courbures et directions principales s'expriment en fonction des dérivées du second ordre de Φ. Plus précisément, elles sont directement fonction de la matrice $W = (w_{i,j}) = \frac{1}{\det(I)^2}\begin{pmatrix} A & D \\ C & B \end{pmatrix}$, ou I, A, B, C sont obtenues par la construction suivante :

- $I = \begin{pmatrix} \langle \Phi_u, \Phi_u \rangle & \langle \Phi_u, \Phi_v \rangle \\ \langle \Phi_u, \Phi_v \rangle & \langle \Phi_v, \Phi_v \rangle \end{pmatrix}$, avec $\Phi_u = \frac{\partial \Phi}{\partial u} = [1, 0, \frac{\partial \phi}{\partial u}]$ et $\Phi_v = \frac{\partial \Phi}{\partial v}$;
- $\mathrm{II} = \begin{pmatrix} \langle N_n, \Phi_{u\,u} \rangle & \langle N_n, \Phi_{u\,v} \rangle \\ \langle N_n, \Phi_{u\,v} \rangle & \langle N_n, \Phi_{v\,v} \rangle \end{pmatrix}$, avec $N = \Phi_u \wedge \Phi_v$, $N_n = \frac{N}{\|N\|}$, $\Phi_{uu} = \frac{\partial \Phi_u}{\partial u}$, $\Phi_{uv} = \frac{\partial \Phi_u}{\partial v}$ et $\Phi_{vv} = \frac{\partial \Phi_v}{\partial v}$;
- $W = I^{-1}\,\mathrm{II}$.

Ici, « générique » veut dire régulier : si Φ est régulière, I est définie positive, et en particulier $\|N\|^2 = \det(I) > 0$.

Compte tenu de la paramétrisation de Φ, et pour ne pas alourdir inutilement les notations, on étudie le problème en projection dans le plan (u, v). Les ridges seront donc vus comme des courbes planes et les points remarquables tels les ombilics ou points pourpres comme des points du plan.

Les courbures principales k_1 et k_2 sont les valeurs propres de W. Avec la convention $k_1 \geq k2$, on a alors $k_1 = \frac{A+D+\sqrt{P_2}}{2\,(\det(I))^{3/2}}$ et $k_2 = \frac{A+D-\sqrt{P_2}}{2\,(\det(I))^{3/2}}$, $P_2 = (A-D)^2 + 4\,B\,C$ étant le discriminant du polynôme caractéristique de W.

Le premier problème pour lequel l'utilisation des méthodes proposées dans ce document est intéressante, est la détermination des ombilics, la connaissance de ces points permettant indéniablement de faciliter la tâche d'un algorithme numérique, par exemple en excluant des petites zones contenant les ombilics. Quelques caractérisations sont connues, par exemple [94] par minimisation de $k_1 - k_2$ ou [88] avec de l'analyse par intervalles, mais les résultats sont difficilement certifiables. Un ombilic étant, par définition, un point ou les courbures principales sont égales, il peut être caractérisé par :

Lemme 6.2. ***FR****-[24] Les définitions suivantes sont équivalentes*

- $(u, v) \in \mathbb{R}^2$ *est un ombilic*
- $P_2(u, v) = 0$*;*
- $A(u, v) - D(u, v) = 0$ et $B(u, v) = 0$ et $C(u, v) = 0$

Sachant que ces ombilics sont en nombre fini dès lors que Φ est régulière, ils sont donc également les solutions réelles du système zéro-dimensionnel $\{P_2 = 0, \frac{\partial P_2}{\partial u} = 0, \frac{\partial P_2}{\partial v} = 0\}$. Si on rassemble toutes les conditions on a que les ombilics sont les solutions réelles de $\mathcal{S}_{\text{omb}} = 0, \mathcal{S}_{\text{omb}} = \{P_2, \frac{\partial P_2}{\partial u}, \frac{\partial P_2}{\partial v}, A - D, B, C\}$.

Sur l'exemple illustrant cette section, ce calcul s'avère « facile », ne prenant pas plus de quelques secondes par l'algorithme proposé au chapitre 3: le système admet 160 racines complexes comptées avec multiplicités, 16 sont réelles et simplement 8 sont dans le pavé unité (domaine de l'étude).

6.1.2.1 La structure algébrique implicite des ridges

Pour caractériser les ridges, une première difficulté est que le signe de b_0 ou b_3 n'est pas bien défini. En dehors des ombilics, si on note d_1 la direction principale associée à k_1 (direction principale maximale), il y a deux vecteurs de directions opposées, notés par la suite y_1 et $-y_1$, permettant l'orientation de d_1. Autrement dit, on peut définir deux coefficients extrêmes $b_0(y_1) = \langle d\,k_1, y_1\rangle$ et $b_0(-y_1) = \langle d\,k_1, -y_1\rangle$, ce qui fait que le signe de b_0 n'est pas défini canoniquement. En particulier, rechercher, par exemple numériquement, les zéros de b_0, c'est à dire déterminer $b_0(d_1)$ en isolant ses changements de signes, nécessite un choix d'orientation cohérent.

Les directions principales peuvent, comme les fonctions k_1 et k_2 s'exprimer formellement. En effet, la direction maximale d_1 (resp. minimale d_2) peut être vue comme étant un vecteur propre de W associée à la valeur propre k_1 (resp. k_2). En développant simplement $\det(W - k_1\,I)$, (resp. $\det(W - k_1\,I)$), on peut choisir pour d_1 (resp. d_2), aussi bien $v_1 = [-2\,B, A - D - \sqrt{2}]$ que $w_1 = [A - D + \sqrt{P_2}, 2\,C]$ (resp. aussi bien $v_2 = [-2\,B, A - D + \sqrt{2}]$ que $w_2 = [A - D - \sqrt{P_2}, 2\,C]$) selon les valeurs prises par A, B, C, D et P_2 (le vecteur ne doit être nul en dehors des ombilics). L'aspect non canonique de la définition des points des ridges (solutions de $b_0 = 0$ et de $b_3 = 0$) se retrouve dans le résultat suivant donnant un critère semi-algébrique pour le choix de la représentation de d_1 ou de d_2 par l'un des vecteur v_1, v_2, w_1, w_2 :

Lemme 6.3. ***FR****-[24]*

- $v_1 = [0, 0] \Leftrightarrow (B = 0 \,\&\, A \geq D)$

- $v_2 = [0,0] \Leftrightarrow (B = 0 \,\&\, A \leq D)$
- $w_1 = [0,0] \Leftrightarrow (C = 0 \,\&\, A \geq D)$
- $w_2 = [0,0] \Leftrightarrow (C = 0 \,\&\, A \leq D)$

On peut par exemple choisir de représenter d_1 par v_1 par défaut et n'utiliser w_1 à la place que lorsque $B = 0 \,\&\, A \geq D$.

En répercutant ces conditions dans la définition initiale des ridges, on peut alors montrer que les ridges bleus et rouges peuvent être vus comme des ensembles semi-algébriques relativement simples :

Lemme 6.4. ***FR**-[24] Il existe des polynômes a, a', b, b' dans $\mathbb{Q}[u,v]$ tels que :*

- *l'ensemble des ridges bleus union $\{v_1 = 0\}$ est l'ensemble des solutions réelles de $a\sqrt{P_2} + b = 0$;*
- *l'ensemble des ridges bleus union $\{w_1 = 0\}$ est l'ensemble des solutions réelles de $a'\sqrt{P_2} + b' = 0$;*
- *l'ensemble des ridges rouges union $\{v_2 = 0\}$ est l'ensemble des solutions réelles de $a\sqrt{P_2} - b = 0$;*
- *l'ensemble des ridges rouges union $\{w_2 = 0\}$ est l'ensemble des solutions réelles de $a'\sqrt{P_2} - b' = 0$;*

les polynômes a, a', b, b' étant définis par

- $a\sqrt{P_2} + b = \langle \text{Numerateur}(d\,k_1), v_1 \rangle$
- $a'\sqrt{P_2} + b' = \langle \text{Numerateur}(d\,k_1), w_1 \rangle$

En cherchant à faire disparaître $\sqrt{P_2}$, on peut remarquer que la clôture algébrique de chacun de ces ensembles est l'union des ridges bleus, des ridges rouges et des ombilics, ce qui est résumé par :

Théorème 6.5. ***FR**-[24] L'union des ridges bleus, des ridges rouges et des ombilics est l'ensemble $\{(u,v) \in \mathbb{R}^2, P(u,v) = 0\}$ avec $P = (a^2 P_2 - b^2)/B = -(a'^2 P_2 - b'^2)/C = 2(a'b - ab')$.*

Si, de plus, on définit la fonction $\text{Sign}_{\text{ridge}}$ comme prenant les valeurs 0 si $ab = a'b' = 0$, -1 si $\left\{ \begin{smallmatrix} ab<0 \\ a'b' \leq 0 \end{smallmatrix} \right.$ ou $\left\{ \begin{smallmatrix} ab \leq 0 \\ a'b'<0 \end{smallmatrix} \right.$, 1 si $\left\{ \begin{smallmatrix} ab>0 \\ a'b' \geq 0 \end{smallmatrix} \right.$ ou $\left\{ \begin{smallmatrix} ab \geq 0 \\ a'b'>0 \end{smallmatrix} \right.$, alors :

- *si $p_2(u,v) = 0$, (u,v) est un ombilic*
- *si $p_2(u,v) \neq 0$ alors*
  - *si $\text{Sign}_{\text{ridge}}(u,v) = +1$, (u,v) appartient à un ridge bleu;*
  - *si $\text{Sign}_{\text{ridge}}(u,v) = -1$, (u,v) appartient à un ridge rouge;*
  - *si $\text{Sign}_{\text{ridge}}(u,v) = 0$, (u,v) est un point pourpre;*

Ainsi, l'étude des ridges (ombilics compris), se ramène exactement à celle d'une courbe plane algébrique et au calcul de signes de polynômes en ses points. Pour un polynôme ϕ de degré total d, les degrés des polynômes apparaissant dans les caractérisations diverses sont bornés par $3\,d - 4$ pour A, B, C, D, $4\,d - 4$ pour $\det(I)$ et $15\,d - 22$ pour P.

Comme évoqué dans la description du problème initial, on peut avoir une caractérisation plus fine des ridges en distinguant les ridges elliptiques des ridges hyperboliques : un ridge est dit elliptique si il correspond à un maximum de k_1 ou un minimum de k_2, il est dit hyperbolique dans tous les autres cas.

La distinction elliptique/hyperbolique est identique dans le principe puisque un ridge bleu (resp. rouge) passe d'hyperbolique à elliptique ou inversement en un « turning point » qui est classiquement défini par l'annulation du Hessien de la courbure principale, c'est à dire lorsque $\mathrm{Hess}(k_1)(d_1, d_1) = 0$ - on dit alors qu'il s'agit d'un « turning point » bleu (resp. $\mathrm{Hess}(k_2)(d_2, d_2) = 0$ - il s'agit alors d'un « turning point » rouge). Le parallèle est flagrant sur la caractérisation des points obtenue dans **FR**-[24]:

Théorème 6.6. ***FR**-[24] Il existe des polynômes $\alpha, \alpha', \beta, \beta'$ de $\mathbb{Q}[u, v]$ tels que :*

- $\mathrm{Numerator}(\mathrm{Hessian}(k_1)(v_1, v_1) = \alpha\sqrt{P_2} + \beta$ *et* $\mathrm{Numerator}(\mathrm{Hessian}(k_1)(w_1, w_1) = \alpha'\sqrt{P_2} + \beta'$*;*
- $\mathrm{Numerator}(\mathrm{Hessian}(k_2)(v_2, v_2) = \alpha\sqrt{P_2} - \beta$ *et* $\mathrm{Numerator}(\mathrm{Hessian}(k_2)(w_2, w_2) = \alpha'\sqrt{P_2} - \beta'$*;*

Si on pose $Q = (\alpha^2 P_2 - \beta^2)/B = -(\alpha'^2 P_2 - \beta'^2)/C = 2(\alpha'\beta - \alpha\beta')$*, alors les « turning points » sont parmi les solutions du système zéro-dimensionnel* $\{P = 0, Q = 0\}$*. Plus précisément, en définissant* $\mathrm{Sign}_{\mathrm{turn}}$ *comme prenant les valeurs* 0 *si* $\alpha\ \beta = \alpha'\ \beta' = 0$, -1 *si* $\begin{cases} \alpha\beta < 0 \\ \alpha'\beta' \leq 0 \end{cases}$ ou $\begin{cases} \alpha\beta \leq 0 \\ \alpha'\beta' < 0 \end{cases}$, 1 *si* $\begin{cases} \alpha\beta > 0 \\ \alpha'\beta' \geq 0 \end{cases}$ ou $\begin{cases} \alpha\beta \geq 0 \\ \alpha'\beta' > 0 \end{cases}$*, alors :*

- *Si* $P_2(u, v) = 0$*, le point est un ombilic;*
- *Si* $P_2(u, v) \neq 0$ *:*
  - *Si* $\mathrm{Sign}_{\mathrm{ridge}}(u, v) = -1$ *alors* (u, v) *est « turning point » bleu;*
  - *Si* $\mathrm{Sign}_{\mathrm{ridge}}(u, v) = +1$ *alors* (u, v) *est « turning point » rouge;*
  - *Si* $\mathrm{Sign}_{\mathrm{ridge}}(u, v) = 0$ (u, v) *est un point pourpre et*
    - *Si* $\mathrm{Sign}_{\mathrm{turn}}(u, v) = -1$ (u, v) *est « turning point » bleu;*
    - *Si* $\mathrm{Sign}_{\mathrm{turn}}(u, v) = +1$ (u, v) *est « turning point » bleu;*
    - *Si* $\mathrm{Sign}_{\mathrm{turn}}(u, v) = 0$ (u, v) *est « turning point » qui est à la fois bleu et rouge (ce qui implique qu'il y ait au moins* $4\ 1/2$ *branches à passer par* (u, v)*;*

Si (u, v) *est un point de* $P = 0$ *qui n'est pas un « turning point » on peut décider si il appartient à une branche parabolique ou elliptique simplement en calculant les signes des polynômes* $\alpha, \alpha', \beta, \beta'$ *en ce point.*

Une remarque est que le système $\{P = 0, Q = 0\}$ définit bien d'autres points que les « turning points » : l'étude de ces points particuliers nécessitera par conséquent le calcul de $\mathrm{Sign}_{\mathrm{turn}}$ et de $\mathrm{Sign}_{\mathrm{ridge}}$ en les zéros de $\{P = 0, Q = 0\}$, c'est à dire l'emploi de l'algorithme le plus général proposé dans le chapitre 3.

6.1.2.2 Un algorithme de tracé certifié

Sur l'exemple utilisé dans cette section, la courbe $P = 0$ est de degré total 84 (43 en chaque variable) et contient 1907 termes avec des coefficients d'au plus 53 chiffres. L'étude de la topologie de cette courbe par les méthodes classiques basées sur un calcul de décomposition cylindrique algébrique [60] est par conséquent exclus, le discriminant de P selon la variable u étant de degré 3594 (le même ordre de grandeur est observable si on échange le rôle des variables u et v) : il est illusoire de prétendre résoudre à l'aveugle et de façon exacte des équations du type $P(\alpha, v) = 0$, α étant une racine du discriminant, et, en particulier, de calculer le nombre de demi-branches passant par chaque point critique de la projection sur l'axe des u (ingrédient principal de ce type d'algorithme).

Toutefois, un petit algorithme très simple permet un tracé certifié et efficace de cette courbe. L'idée est de garantir ce que l'on peut voir à l'écran c'est à dire à l'échelle du pixel, avec la convention suivante : un pixel est supposé être un carré - il sera allumé si la courbe coupe au moins un de ses cotés opposés, éteint sinon.

Schématiquement, l'algorithme réalise les opérations suivantes (plus de détails dans [24]) :

- On découpe le domaine d'étude $[0,1] \times [0,1]$ en pavés $[\frac{i}{n}, \frac{i+1}{n}] \times [\frac{j}{n}, \frac{j+1}{n}]$, n étant la résolution désirée i, j étant des entiers variant entre 0 et $n-1$;
- On calcule les zéros réels de $P(\frac{i}{n}, v)$ dans $[0,1]$ par la méthode proposée au chapitre 2 et on raffine les intervalles d'isolation de sorte à ce qu'ils ne chevauchent aucun intervalle de la forme $[\frac{j}{n}, \frac{j+1}{n}]$;
- On réalise le même calcul pour $P(u, \frac{j}{n})$ si bien que l'on peut alors déterminer quels pavés $[\frac{i}{n}, \frac{i+1}{n}] \times [\frac{j}{n}, \frac{j+1}{n}]$ seront allumés.

Notons qu'avec ces spécifications, on ne distinguera pas à l'écran les composantes connexes strictement contenues dans un pixel, sauf si l'on connaît la distance minimale séparant deux points critique de la projection sur l'axe des u (ou sur l'axe des v). Malgré la taille de P, cet algorithme s'avère très efficace en pratique puisqu'il ne faut que quelques secondes pour obtenir un tracé de 1024x1024 pixels sur l'exemple illustrant cette section.

Constatant l'efficacité de cette solution, on a alors pu utiliser la version « étendue » de l'algorithme d'isolation du chapitre 2 et calculer en plus les signes de $a', a, b', c', \alpha, \alpha', \beta, \beta'$ en les points $(\frac{i}{n}, \gamma)$ (resp. $(\gamma, \frac{j}{n})$) ou α est un zéro de $P(\frac{i}{n}, v)$ (resp. $P(u, \frac{j}{n})$) et ainsi caractériser la couleur (bleue ou rouge) ainsi que le type (elliptique ou hyperbolique) de chaque ridge, ce qui donne, en projection dans le plan (u, v). L'image qui suit est un tracé certifie 1024x1024 en projection dans le plan (u, v), calculé avec les outils mentionnés dans cette section. Les ridges bleus elliptiques sont en vert, les ridges rouges elliptiques sont en jaune.

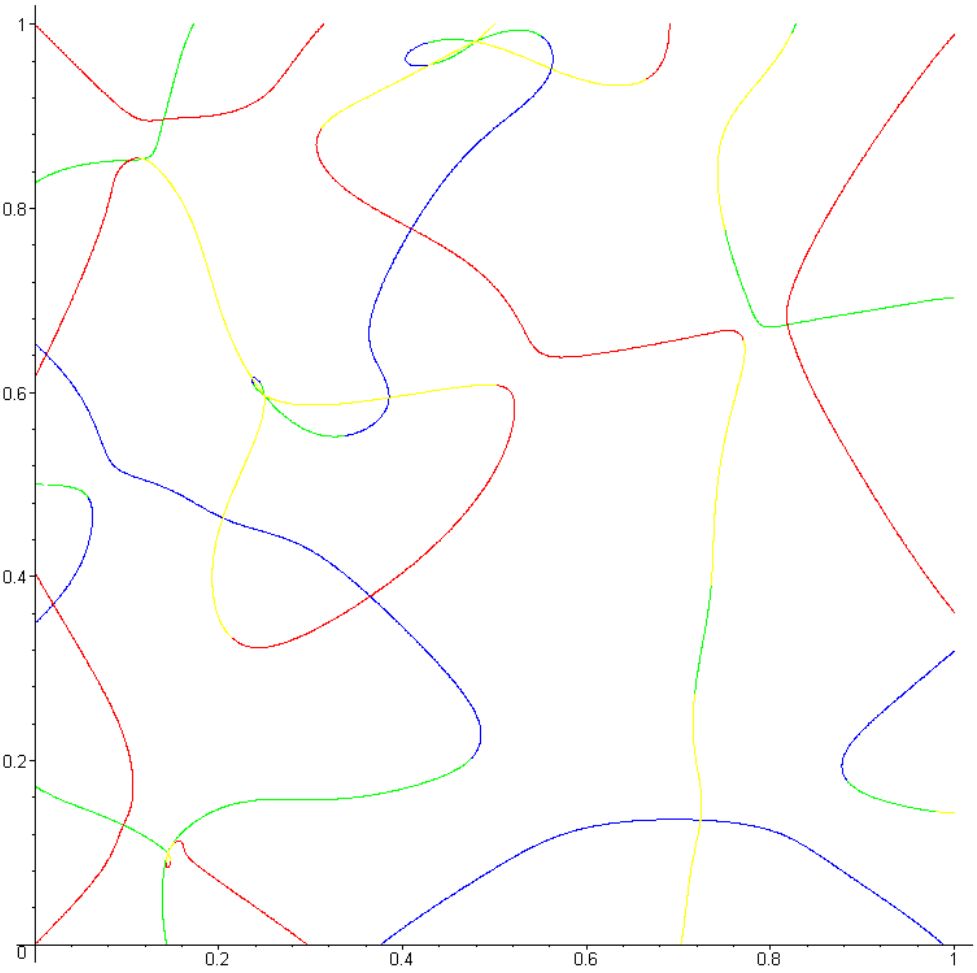


Figure 6.5. Projection colorée des ridges de l'exemple

6.1.2.3 Calcul de la topologie des ridges

Le calcul de la topologie d'une courbe telle que $P = 0$ pour l'exemple utilisé dans cette section est clairement un « challenge » à la vue de son degré (de degré total 84, de degré 43 en chacune de ses variables), de sa densité (1907 termes) et de la taille des coefficients de P (jusqu'à 53 chiffres) et de son irréductibilité.

La stratégie usuelle [60] est basée sur un calcul de décomposition cylindrique algébrique dont on peut résumer les étapes de la façon suivante :

- [A] calcul des points critiques de la projection sur l'axe v, incluant les points singuliers : les coordonnées v de ces points sont par exemple données par le discriminant de P relativement à la variable u; On note $\mathcal{C}_u$ ces points et $\mathcal{D}_u$ leur projection sur l'axe v.

- [B] calcul des intersections avec le domaine $[0, 1] \times [0, 1]$ pour déterminer les branches de courbes entrantes et sortantes. On note $\mathcal{C}_{u,\infty}$ ces points et $\mathcal{D}_{u,\infty}$ leurs projections de ces points sur l'axe v.

- [C] quitte à effectuer un changement de variable affine on suppose que $\mathcal{C}_u \cap \mathcal{C}_{u,\infty} = \emptyset$ et que, pour tout $v_0 \in \mathcal{D}_u \cup \mathcal{D}_{u,\infty}$, $P(u, v_0)$, ait une unique solution (u_0, v_0) telle que $(u_0, v_0) \in \mathcal{C}_u \cup \mathcal{C}_{u,\infty}$. Ceci peut se vérifier en calculant les multiplicités des solutions de $P(u, v_0)$ qui ne doit pas contenir plus d'un point multiple, et en vérifiant que $\mathcal{D}_u \cap \mathcal{D}_{u,\infty} = \emptyset$.

- [D] calcul du nombre de 1/2 branches en les points de $\mathcal{C}_u$.

- [E] supposant que $\mathcal{D}_u \cup \mathcal{D}_{u,\infty} = \{v_0, ..., v_k\}$, on sait alors que les branches au dessus de tout intervalle $]v_i, v_{i+1}[$ ne s'intersectent pas et ne rencontrent pas les bornes du domaine d'étude. On peut alors les numéroter, par exemple en désignant par $b_{i,j}$ la branche de courbe située au dessus de l'intervalle $]v_i, v_{i+1}[$ et contenant la j-ème racine dans $[0, 1]$ de tout polynôme $P(u, \alpha)$, $\alpha \in]v_i, v_{i+1}[$. On ordonne également les points $p_{i,j}$ solutions de $P(u, v_i) = 0$ dans $[0, 1]$.

- [F] connaissant la numérotation des branches $b_{i,j}$ au dessus de $]v_i, v_{i+1}[$ ainsi que le nombre de 1/2 branches de connectant aux points $p_{i,j}$, on obtient alors le graphe topologique de $P = 0$ dans le pavé $[0, 1] \times [0, 1]$

Ce schéma de calcul est classique et proche d'être optimal si on veut obtenir une sortie exploitable, nous nous y tiendrons donc. Par contre, les méthodes de calcul employées, y compris dans les variantes les plus abouties telles que [60] ne permettent aucunement de traiter des exemples de la taille de celui qui illustre cette section. En effet, un point clé est le calcul du nombre de 1/2 branches se connectant à un point de $\mathcal{C}_u$, la mise en « position générique » (étape [C]) pouvant être relativement facilement contournée, et le traitement des points de $\mathcal{C}_{u,\infty}$, bien qu'étant un écueil pour l'implantation, rendant relativement trivial à maîtriser.

Dans [60], l'idée est d'exploiter au mieux les propriétés des suites de Sturm-Habicht et leur relations avec les sous-résultants, montrant alors que l'unique véritable difficulté est de résoudre les équations $P(u, v_i)$, c'est à dire isoler les solutions réelles et en calculer les multiplicités. Sur notre exemple, ceci revient à résoudre des polynômes de degré 43 dont les coefficients dépendent des zéros réels du discriminant de P selon la variable u qui est de degré 3594, ce qui n'est pas imaginable avec les stratégies proposées dans [60](suites de Sturm-Habicht), en tout cas pas de façon fiable (utilisation d'arithmétique flottante).

Le problème de l'étude des ridges est très particulier puisque l'on sait caractériser complètement les points singuliers, en tout cas avec nos hypothèses, celles-ci ne pouvant être que de 3 types :

- ombilics de type 1-R, c'est à dire avec simplement 2 1/2 branches s'y connectant;

- ombilics de type 3-R, c'est à dire avec 6 1/2 branches s'y connectant;
- points pourpres, avec 4 1/2 branches s'y connectant.

L'idée est de réussir à séparer et calculer ces points de sorte à éviter l'étude des fibres $P(u, v_i)$ pour calculer la topologie de $P = 0$. Les résultats de [24] permettent de décomposer et de caractériser les points de $\mathcal{C}_u$:

Théorème 6.7. ***FR**-[24] Les ombilics 1-R, 3-R et les points pourpres peuvent être caractérisés par les valeurs de polynômes en les racines d'un système zéro-dimensionnel. Précisément, si D_3 désigne le discriminant de la cubique des dérivées 3-ièmes de P à un ombilic et si $\mathcal{S} = \{P = \frac{\partial P}{\partial u} = 0\}$, alors :*

- *Les ombilics 1-R sont les solutions réelles de $\mathcal{S}_{1R} = \mathcal{S} \cap \{\frac{\partial P}{\partial v} = 0\} \cap \{P_2 = 0\} \cap \{\delta(D_3) < 0\}$;*
- *Les ombilics 3-R sont les solutions réelles de $\mathcal{S}_{3R} = \mathcal{S} \cap \{\frac{\partial P}{\partial v} = 0\} \cap \{P_2 = 0\} \cap \{\delta(D_3) > 0\}$;*
- *Les points pourpres sont les solutions réelles de $\mathcal{S}_p = \mathcal{S} \cap \{\frac{\partial P}{\partial v} = 0\} \cap \{P_2 \neq 0\}$*
- *Les points critiques de la projection sur l'axe v sont les solutions réelles de $\mathcal{S}_c = \mathcal{S} \cup \{\frac{\partial P}{\partial v} \neq 0\}$*
- *L'ensemble $\mathcal{C}_u$ est l'union des 4 ensembles disjoints $\mathcal{S}_{1R}, \mathcal{S}_{3R}, \mathcal{S}_p$ et $\mathcal{S}_c$.*

Cette formulation « simplifiée » ne tient pas compte des nombreuses équations pouvant être rajoutées dans la caractérisation des divers ensembles permettant de faire chuter nettement le degré des systèmes. Par exemple, les points pourpres vérifient tous $a = a' = b = b' = 0$. Elle met également la structure « gigogne » des idéaux à étudier pour la résolution des systèmes mis en jeux. Si on pose $I = \langle P, \frac{\partial P}{\partial u} \rangle$, alors $\mathcal{S}_c = \boldsymbol{V}(I : \left(\frac{\partial P}{\partial v}\right)^{\infty}) \cap \mathbb{R}^2$. On peut alors poser $I' = I + \langle \frac{\partial P}{\partial v} \rangle$ pour obtenir $\mathcal{S}_p = \boldsymbol{V}(I' : P_2^{\infty}) \cap \mathbb{R}^2$ et enfin poser $I'' = I' + \langle P_2 \rangle$ pour obtenir $\mathcal{S}_{1R} \cup \mathcal{S}_{3R} = \boldsymbol{V}(I'') \cap \mathbb{R}^2$. Ce type de décomposition a pu être exploité par Jean-Charles Faugère pour produire des bases de Gröbner pour les idéaux I, I' et I'' et éviter de gros calculs en décomposant l'ensemble des points à étudier. En particulier, si on élimine u dans chacun de ces idéaux, on obtient une décomposition du discriminant de P par rapport à u comme produit de 3 facteurs.

Une fois ces bases de Gröbner obtenues, il ne reste plus qu'à appliquer l'algorithme du chapitre 3 pour obtenir des pavés de $\mathbb{R}^2$, raffinables à volonté et isolant tous les points de $\mathcal{C}_u$. Notons au passage que la décomposition du problème permet de supposer également que les pavés obtenus ne s'intersectent pas si ils sont suffisamment raffinés.

Une fois obtenus ces points sous forme de RUR, il est facile de vérifier que la courbe est en position générique et de les projeter sur l'axe v. On a alors tous les éléments pour calculer la topologie de la courbe **FR**-[25] :

- [A] Isolation les points « remarquables » : ceux de $\mathcal{C}_u$ par la méthode ci-dessus, ceux de $\mathcal{C}_{u,\infty}$ simplement en résolvant $P(0, v) = 0, P(u, 0) = 0, P(1, v) = 0, P(u, 1) = 0$; On suppose les pavés isolant ces points tous distincts quitte à les raffiner;
- [B] Régularisation des boites d'isolation. On raffine les boites d'isolation jusqu'à ce la courbe intersecte leur bord en le bon nombre de points : 2 pour un ombilic $1R$, 6 pour un ombilic $3R$, 4 pour un point pourpre, 2 pour un point critique et 1 pour un point sur le bord du domaine. Pour simplifier les calculs qui suivent, on les raffine jusqu'à ce que tous les points d'intersection soient sur les cotés de la boite parallèle à l'axe des u. La courbe étant en position générique, on peut également supposer que la projection de ces boites sur l'axe v donne des intervalles ne s'intersectant pas : ils représentent alors les points v_i de l'algorithme général de CAD.

- [C] Calcul d'une approximation des points réguliers des fibres : pour tout intervalle sur l'axe v qui est projection d'une boite d'isolation calculée ci-dessus, on choisi un rationnel q arbitrairement dans cet intervalle, on résout $P(u,\, q) = 0$ dans $[0,\, 1]$ et on marque les solutions qui sont dans les boites d'isolation de [B], on est alors sûrs que les solutions non marquées sont des approximations de solutions simples de $P(u,\, \alpha)$, α racine du discriminant de P par rapport à u.

- [D] A ce stade, on a alors une approximation certifiée des fibres $P(u,\, v_i)$, les points étant représentés par des pavés, et on connaît le nombre de $1/2$ branches se connectant à chacun de ces points. On peut donc appliquer les étapes [E] et [F] de la CAD et conclure.

Comme remarqué dans **FR**-[25], cette méthode permet même d'éviter de forcer la courbe à être en position générique, en tout cas pour le calcul de topologie des ridges, le fait d'avoir plus d'un point multiple dans une fibre ne compliquant que l'implantation. En pratique, les calculs sont assez longs, il faut compter environ $1/2$ heure sur une machine performante du moment pour obtenir le graphe topologique complet et coloré sur l'exemple utilisé pour illustrer cette section.

A cause du nombre important de points « remarquables », le graphe topologique obtenu est très proche, en apparence, du tracé certifié donné à la section précédente, en tout cas pour la résolution visible sur une page imprimée.

6.2 Systèmes de dimension positive

Comme pour le cas zéro-dimensionnel, deux applications ont été sélectionner pour illustrer l'utilisation des algorithmes proposés dans ce document. Pour la première, le problème d'interpolation de Birkhoff, on peut réutiliser une formulation déjà bien aboutie [58], alors que pour la deuxième (synthèse de bancs de filtres), il aura fallu remonter relativement haut dans le problème pour trouver une modélisation adaptée.

6.2.1 Le problème d'interpolation de Birkhoff

Étant donnée une fonction $f\colon \mathbb{R} \to \mathbb{R}$, le problème d'interpolation de Birkhoff consiste à décider si la connaissance des valeurs de f ainsi que de certaines de ses dérivées en un ensemble de points fixé suffit à déterminer un polynôme p, de degré fixé, interpolant f.

L'interpolation d'une fonction inconnue $f\colon \mathbb{R} \to \mathbb{R}$ par un polynôme en connaissant les valeurs de f et de certaines de ses dérivées en quelques points de $\mathbb{R}$ est un problème fondamental en analyse numérique et en théorie de l'approximation.

Les deux problèmes d'interpolation les plus connues sont l'interpolation de La-grange et de Hermite. Dans le premier cas, si on suppose connues les valeurs de f en n points $x_0 < ... < x_n$ de $\mathbb{R}$, la formule de Lagrange montre qu'il existe un polynôme de degré inférieur ou égal à n interpolant f aux points x_i. Le problème d'interpolation de Hermite généralise celui de Lagrange en ce sens que des informations concernant les dérivées de f peuvent être inclues. Si $x_1 < ... < x_n$ sont des points de $\mathbb{R}$ fixés et $\nu_1, ..., \nu_n$ des entiers positifs, le problème d'interpolation de Hermite est résolu lorsque l'on peut trouver un polynôme P (unique) de degré inférieur ou égal à $N = \nu_1 + ... + \nu_n - 1$ tel que pour tout $k \in \{1, ..., n\}$ et $j \in \{0, ..., \nu_{k-1}\}$ on ait $f^{(j)}(x_k) = P^{(j)}(x_k)$.

Le problème d'interpolation de Birkhoff est quant à lui encore plus général en ce sens qu'aucune contrainte à priori n'est imposée tant sur le nombre ou l'ordre des dérivées que sur le nombre de points considérés. On se donne un ensemble de nombres réels $\chi = \{x_1, ..., x_n\}$ tels que $x_1 < \cdots < x_n$, r un entier, et on suppose que $\mathcal{I} \subset \{1, ..., n\} \times \{0, ..., r\}$ est l'ensemble des couples (i, j) tels que $f_{i,j} = f^{(j)}(x_i)$ est connu. Le problème d'interpolation de Birkhoff est de déterminer l'existence et l'unicité d'un polynôme $Q \in \mathbb{R}[X]$ de degré borné par r tel que $\forall (i,\, j) \in \mathcal{I}$, $Q^{(j)}(x_i) = f_{i,j}$.

La question peut certainement être abordée de multiple façons, mais nous ne retiendrons que la modélisation proposée dans [58] qui est certainement celle qui est la plus adaptée au calcul algébrique. Dans [58], l'auteur propose une méthode pour calculer l'ensemble $\mathcal{I}$ pour n et r fixés, se ramenant à un problème matriciel « simple ».

Considérons la matrice $\mathcal{E} = (e_{i,j})$ ayant n lignes et $r+1$ colonnes, dont les entrées sont des 0 et des 1 et telles que $e_{i,j} = 1$ si et seulement si $(i, j) \in \mathcal{I}$. Résoudre le problème, revient à dire que les coefficients d'un polynôme d'interpolation Q sont solutions d'un système linéaire dont la matrice associée sera désormais notée $\mathcal{M}_{\mathcal{E}}$. Cette matrice est paramétrée par χ et sa forme est donnée par $\mathcal{E}$. En résumé, le problème consiste à déterminer les matrices $\mathcal{E}$ pour lesquelles le déterminant de $\mathcal{M}_{\mathcal{E}}$ est non nul pour tout χ, auquel cas la matrice $\mathcal{E}$ est dite *poised*.

Exemple 6.8. Supposons $n = 4$ et $r = 3$ et considérons la matrice:

$$\mathcal{E} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Soit $Q(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$, la propriété « $Q^{(j)}(x_i) = f_{i,j}$ si et si seulement si $e_{i,j} = 1$ » induit le système linéaire :

$$\begin{cases} a_0 + a_1 x_1 + a_2 x_1^2 + a_3 x_1^3 &= f_{1,1} \\ a_1 + 2a_2 x_1 + 3a_3 x_1^2 &= f_{1,2} \\ a_1 + 2a_2 x_3 + 3a_3 x_3^2 &= f_{3,2} \\ 2a_2 + 6a_3 x_2 &= f_{3,2} \end{cases}$$

dont la matrice est:

$$\begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 0 & 1 & 2x_1 & 3x_1^2 \\ 0 & 1 & 2x_3 & 3x_3^2 \\ 0 & 0 & 2 & 6x_2 \end{pmatrix}$$

Ainsi, le problème d'interpolation admet une unique solution si et seulement si le polynôme $12x_3x_2 + 6x_1^2 - 12x_2x_1 - 6x_3^2$ ne s'annule pas quels que soient les réels x_1, x_2, x_3 tels que $x_1 < x_2 < x_3$.

Dans [58], l'auteur montre que l'on peut réduire le problème en remplaçant le déterminant de $\mathcal{M}_{\mathcal{E}}$ l'un de ses facteurs, puis. en remplaçant $(x_1, ..., x_n)$ par $(x_1, x_1 + t_1^2, ..., x_1 + t_1^2 + ... + t_{n-1}^2)$ il obtient alors ce polynôme homogène en $(t_1, ..., t_{n-1})$. Enfin, en posant $t_1 = 1$, il montre que le problème se réduit à l'étude d'une variété réelle définie par un unique polynôme $P \in \mathbb{R}[t_2, ..., t_{n-1}]$ dont on veut tester si elle admet ou non des points réels avec des coordonnées non nulles, ce qui peut en fin de compte se traduire par une unique égalité $P^2 + (T x_1 x_2 x_3 - 1)^2 = 0$, T étant une nouvelle variable.

Les cas $n = 2$, $r \in \{1, ..., 6\}$, $n = 3$, $r \in \{1, 2, 3\}$ and $n = 4$, $r \in \{1, ..., 4\}$ sont résolus dans [58], en utilisant des techniques voisines de celles utilisées pour le calcul d'une Décomposition Cylindrique algébrique.

Le cas $n = 5$ and $r = 4$ est résolu dans **FR**-[113], mais pas automatiquement, certains cas ayant été étudiés « à la main ». Il est donc maintenu comme problème ouvert dans [133]. Les autres cas sont soit triviaux, soit pas abordables par le biais de cette modélisation par les matrices *poised*, compte tenu du nombre de cas à étudier qui est $\binom{n(r+1)}{r+1}$.

Ce problème a donc servi naturellement à motiver l'étude des méthodes de points critiques présentées dans le chapitre 4 et surtout a permis de se faire une idée sur le choix de stratégie de calcul à adopter. En effet, cette étude est consécutive à l'optimisation de l'algorithme [13] basé sur l'utilisation de déformations infinitésimales formelles et présentée dans **FR**-[118] (voir chapitre 4). Sur la résolution du problème d'interpolation de Birkhoff, on constate, en pratique, un avantage significatif de ce type de méthodes sur des algorithmes dérivés de la décomposition cylindrique algébrique, inversant donc les constatations de [68].

Il faut constater que les hyper-surfaces traitées sont de faibles degrés et dépendent de relativement peu de variables. Le fait que certains calculs n'aient pu être effectués est donc un échec cuisant. Cette étude a donc également motivé la mise au point de l'algorithme proposé dans **FR**-[7] (voir chapitre 4) dont la principale particularité est de ne jamais avoir recours à des déformations infinitésimales, mais également d'être prévu pour traiter directement les variétés définies par un un système d'équations sans utiliser l'artifice (aux effets pratiques désastreux) qui consiste à prendre la somme des carrés des équations pour se ramener à l'étude d'une seule équation.

Dans le tableau suivant, l'algorithme CRITICALHYP est une instance probabiliste de **FR**-[118] ou le calcul de RUR est effectué par diverses interpolations spécialement optimisées et récupérant d'ailleurs certains principes utilisés dans [56], l'algorithme CRITICAL est celui de **FR**-[7] :

Hypersurface	CRITICALHYP	CRITICAL
Birk.3-1	5.6	0,08
Birk.3-2	5.2	0,13
Birk.3-3	32	0,37
Birk.3-4	46	0,18
Birk.3-5	116	0,46
Birk.3-6	149	0,86
Birk.3-7	115	0,72
Birk.3-8	3927	7,11
Birk.3-9	2945	7,88
Birk.3-10	18843	8,04
Birk.3-11	26536	8,88
Birk.3-12	17508	10,01
Birk.3-13	∞	9,26
Birk.3-14	∞	67
Birk.3-15	∞	83

L'écart entre les deux catégories de méthodes est assurément accentué avec [126] mais de toutes façons, le problème de Birkhoff pour $n=5$ et $r=4$ est résolu de façon automatique depuis **FR**-[7].

6.2.2 Synthèse de bancs de filtres de dimension 2 non séparables

Note : cette section décrit les résultats publiés dans **FR**-[46] et objet du brevet **FR**-[45].

Un signal modélise une grandeur physique variant dans le temps (son, image). Nous supposerons ici qu'un signal est la donnée d'une suite discrète d'informations, ce qui correspond à la modélisation standard en pratique (transmission de données, compression, ...). Sous certaines conditions on peut échantillonner un signal (de nature continue $x(t)$) sans perdre d'information. C'est le cas, par exemple, des signaux à bande limitée (transformée de Fourier à support compact).

Un signal sera par conséquent la donnée d'une suite d'éléments $X=(x_i)_{i\in\mathbb{Z}}$ de $\mathbb{R}^n$ que nous supposerons sommable (i.e. : $\sum_{i=-\infty}^{+\infty}|x_i|<\infty$) et de carré sommable. Ceci n'est pas, en pratique, une contrainte, les volumes d'informations traités restant forcément finis.

Cette représentation , dite temporelle, admet une forme duale (*représentation fréquentielle*). En théorie, la *représentation fréquentielle* d'un signal en temps continu (suffisamment régulier) est la donnée de sa transformée de Fourier. Dans le cas d'un signal discret de représentation temporelle $X=(x_i)_{i\in\mathbb{Z}}$, celle-ci peut se déduire simplement de la transformée en Z de X :

$$\hat{X}(z) = \sum_{k=-\infty}^{+\infty} x_{-k} z^{-k}$$

(La *représentation fréquentielle* de X est donnée par la fonction $f \longmapsto \hat{X}(e^{2i\pi f})$).

Lorsque l'on veut, par exemple, compresser l'information contenue dans un signal X donné, on applique une transformation, de préférence réversible, définie sous le terme générique de filtrage.

Une opération de filtrage consiste à transformer un signal X en un autre signal. Mathématiquement, ceci se définit par la convolution de X et d'une mesure de Radon. Dans notre cas (signaux discrets) il s'agit simplement de la convolution de deux suites et nous nous limiterons même au cas de filtres linéaires à support fini, traduisant ainsi l'opération de filtrage par la convolution de X et d'un polynôme de Laurent.

La transformée de Fourier d'une convolution étant égale au produit des transformées de Fourier (pour des fonctions suffisamment régulières), lorsque l'on considère les représentations fréquentielles, l'image de $\hat{X}(z)$ par un filtre linéaire à support fini est donc de la forme :

$$\hat{X}(z) H(z)$$

où $H(z)$ est un polynôme de Laurent.

Une catégorie particulière de filtres nous intéresse : les filtres *passe-bande* et en particulier les filtres *passe-haut* et *passe-bas*.

Un filtre *passe-bas* (resp. *passe-haut*) est une transformation permettant d'atténuer la contribution des fréquences élevées (resp. faibles). Rappelons que l'expression fréquentielle d'un signal est une fonction périodique ($f \longmapsto \hat{X}(e^{2i\pi f})$), et que nous entendons par *basse fréquence* (resp. *haute fréquence*) les valeurs de f proches de $0+n$, $n \in \mathbb{Z}$ (resp. $1/2+n$, $n \in \mathbb{Z}$).

Pour tout signal $X = (x_i)_{i\in\mathbb{Z}}$, nous noterons $X\downarrow 2 = (y_i = x_{2i})_{i\in\mathbb{Z}}$ (sous-échantillonnage) et $X\uparrow 2 = (y_{2i} = x_i \ , \ y_{2i+1} = 0)_{i\in\mathbb{Z}}$ (sur-échantillonnage).

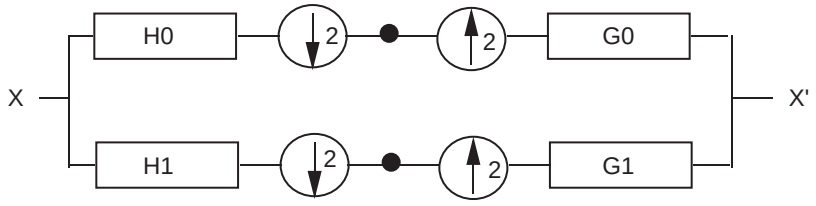


Figure 6.6. Décomposition par les filtres *passe bande*

La figure 6.6 résume une opération de filtrage élémentaire (dans le cas de signaux uni-dimensionnels), pour laquelle le but est de trouver des filtres G_i, H_i tels que $\hat{X}' = \hat{X}$.
L'expression algébrique résumant le schéma étant :

$$\begin{aligned} \hat{X}' = \ & 1/2 G_0(Z)[\hat{X}(Z)H_0(Z) + \hat{X}(-Z)H_0(-Z)] \\ & + 1/2 G_1(Z)[\hat{X}(Z)H_1(Z) + \hat{X}(-Z)H_1(-Z)] \end{aligned} ,$$

cette condition s'exprime simplement en annulant le coefficient de $\hat{X}(-Z)$ dans :

$$\begin{aligned} \hat{X}' = \ & 1/2 \hat{X}(Z)[G_0(Z)H_0(Z) + G_1(Z)H_1(Z)] + \\ & 1/2 \hat{X}(-Z)[G_0(Z)H_0(-Z) + G_1(Z)H_1(-Z)] \end{aligned}$$

En posant :

$$G_0(Z)H_0(-Z) + G_1(Z)H_1(-Z) = 0 \, , \tag{6.3}$$

nous obtenons, à un facteur retard près (terme de la forme Z^{-d} , d impair) :

$$G_1(Z) = H_0(-Z) \ \ H_1(Z) = -G_0(-Z) \tag{6.4}$$

En réalisant une décomposition *polyphase* des filtres H_i et G_i, c'est à dire en posant :

$$G_i(Z) = G_{i0}(Z^2) + Z\,G_{i1}(Z^2)\ ,\ i = 0,1$$

$$H_i(Z) = H_{i0}(Z^2) + Z\,H_{i1}(Z^2)\ ,\ i = 0,1$$

on peut exprimer la relation (6.3) sous forme matricielle :

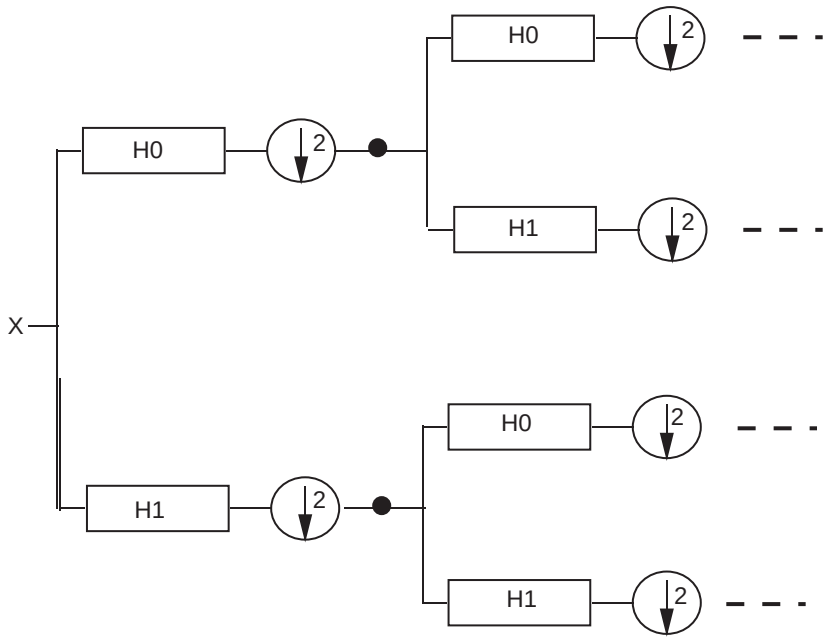
$$\begin{bmatrix} G_{00} & G_{01} \\ G_{10} & G_{11} \end{bmatrix} \begin{bmatrix} H_{00} & H_{10} \\ H_{01} & H_{11} \end{bmatrix} = I\,d\,Z^{-d'}$$

Si on impose que la base sur laquelle on décompose le signal (expression obtenue après filtrage et sous échantillonnage) est *orthogonale*, l'expression précédente conduit alors à une égalité de la forme :

$$H_0(Z) = Z^L G_0(1/Z)$$

Le problème décrit par la figure 6.6 se résume par conséquent à l'étude d'un unique polynôme de Laurent, par exemple H_0 (*synthèse de filtre*).

La décomposition d'un signal $\hat{X}$ par le biais de filtres *passe-haut* et *passe-bas* (figure 6.6), peut être itérée comme décrit par la figure 6.7, on définit ainsi un *banc de filtres itérés*.



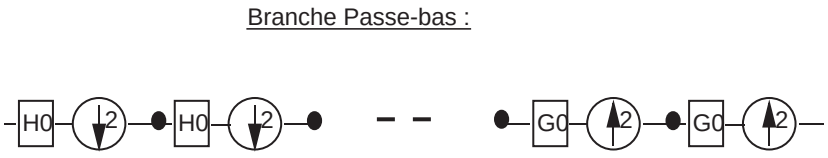


Figure 6.7. Itération de filtres *passe bande*

Par des lois classiques de permutations entre filtrages et sous/sur-échantillonnages, il est possible de modéliser chaque branche par une composée plus simple. Par exemple la branche la plus *haute* (ne faisant intervenir que les filtres H_0 et G_0) peut être résumée par un diagramme identique à celui de la figure 6.7, où, dans le cas de k itérations, le filtre H s'exprimera simplement par :

$$H(z) = H_0(Z)H_0(Z^2)...H_0(Z^{2k-1})$$

La fonction limite du processus itéré établit alors le lien entre l'étude du banc de filtres itérés et la théorie des *ondelettes*.

6.2.2.1 Le problème

On cherche à synthétiser un filtre *passe-bas* générant une base d'ondelettes orthogonale. Sans entrer dans les détails, disons que ceci revient à définir un polynôme (filtre) H_0, satisfaisant un certain nombre de relations algébriques et optimum pour un certain critère que nous préciserons selon selon le cas étudié.

Par exemple, dans le cas de signaux $1D$ (à valeurs dans $\mathbb{R}$), les contraintes algébriques caractérisant les filtres générant une base d'ondelettes orthogonale permettent d'établir une paramétrisation des filtres H_0 sous la forme :

$$H_0(z) = \begin{bmatrix} \cos\theta_0 & \sin\theta_0 \\ -\sin\theta_0 & \cos\theta_0 \end{bmatrix} \left(\prod_{k=1}^{K-1} \begin{bmatrix} 1 & 0 \\ 1 & z^2 \end{bmatrix} \begin{bmatrix} \cos\theta_k & \sin\theta_k \\ -\sin\theta_k & \cos\theta_k \end{bmatrix} \right) \begin{bmatrix} 1 \\ z \end{bmatrix} \tag{6.5}$$

La décomposition des solutions sous forme d'une telle *cascade* est largement reconnue par les spécialistes, à cause de la simplicité de la paramétrisation, pour l'efficacité des implantations hardware qui l'utilisent, et enfin et surtout parce que c'est une paramétrisation complète de l'espace sur lequel on effectue l'optimisation, sans aucune redondance.

Nous nous intéressons ici au cas des signaux $2D$ (bidimensionnels) pour lesquelles les choses se présentent de façon un peu plus complexe.

La décomposition d'un signal $\hat{X}$ par le biais de filtres (polynômes de Laurent en deux variables) *passe-bande* itérés s'effectue de la même manière que dans le cas $1D$.

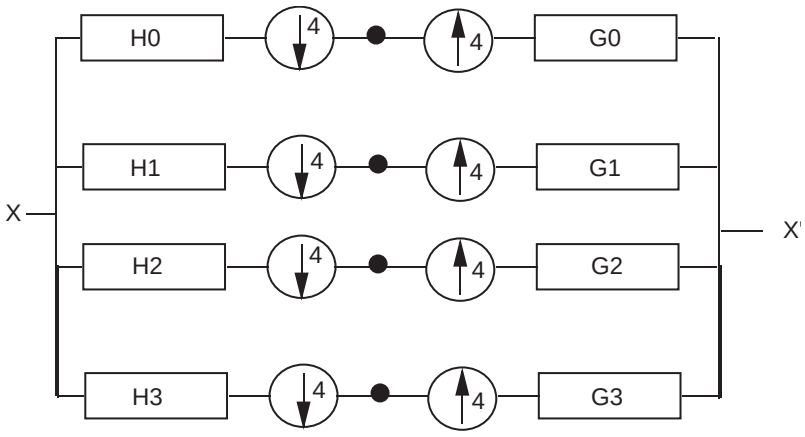


Figure 6.8. Décomposition d'un signal 2D

Si l'on se place dans le cadre d'un échantillonnage de type $2I$ (système à 4 sous bandes séparable), la figure 6.8 résume une situation que nous pouvons décrire de la façon suivante :

- La représentation temporelle d'un signal $2D$ est la donnée d'une suite $X = (x_{i,j})_{(i,j)\in\mathbb{Z}^2}$ d'éléments de $\mathbb{R}^2$
- Sa représentation fréquentielle est alors déductible à partir de l'expression :

$$\sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} x_{i,j} z_0^{-i} z_1^{-j}$$

- Les sous-échantillonnages consistent alors à prendre simplement les termes de la forme $x_{2i,2j}$ dans l'expression de $\hat{X}$.

- Dans le cas $2D$, la notion de filtre *passe-bas* est similaire à celle du cas $1D$ (réponse en fréquence faible sauf autour des points entiers), la définition d'un filtre *passe-haut* n'ayant, en général, pas de sens précis.

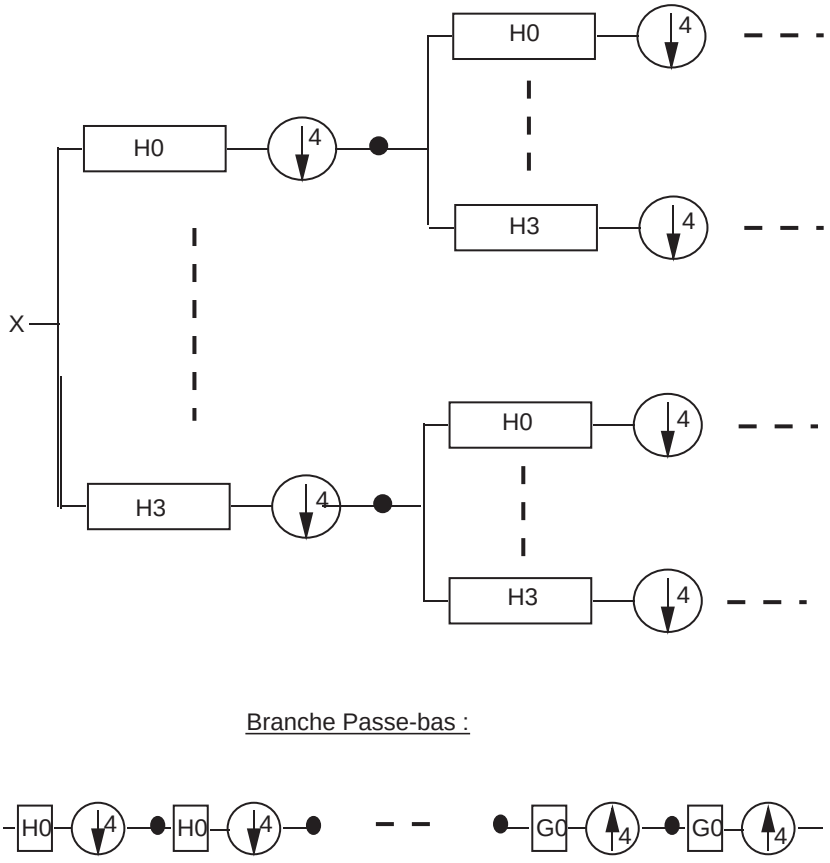


Figure 6.9. Filtres itérés

Comme dans le cas $1D$, l'expression fréquentielle du signal filtré s'exprime par une égalité de la forme :

$$\hat{X}' = A_1 X(z_1, z_2) + A_2 X(-z_1, z_2) + A_3 X(z_1, -z_2) + A_4 X(-z_1, -z_2)$$

et la *reconstruction parfaite* impose alors

$$A_2 = A_3 = A_4 = 0$$

Toujours comme dans le cas $1D$, on peut itérer le processus (voir figure 6.9), définissant ainsi un filtre H_0, équivalent à la branche *passe-bas* et s'exprimant sous la forme d'un polynôme dépendant de $(z_1, z_2, z_1^{-1}, z_2^{-1})$ et de k (nombre d'itérations), dont la limite, lorsque $k \to \infty$, établit le lien avec la théorie des *ondelettes*.

Dans le cas $2D$, nous n'avons plus de description complète de l'espace (simplement paramétrée) des filtres à étudier et nous limitons l'étude à celle d'une cascade de type **Vetterly-Kovacevic PL** (voir [75]) présentant un certain nombre de *bonnes* propriétés :

- orthogonalité de la base de décomposition.
- filtres (i.e. les fonctions de base) symétriques ou antisymétriques par rapport au centre de leur support.

On note:

$$R = \begin{bmatrix} \cos\alpha_i & -\sin\alpha_i & 0 & 0 \\ \sin\alpha_i & \cos\alpha_i & 0 & 0 \\ 0 & 0 & \cos\beta_i & -\sin\beta_i \\ 0 & 0 & \sin\beta_i & \cos\beta_i \end{bmatrix} \tag{6.6}$$

$$W = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \tag{6.7}$$

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{6.8}$$

$$D(z_1, z_2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & z_1 & 0 & 0 \\ 0 & 0 & z_2 & 0 \\ 0 & 0 & 0 & z_1 z_2 \end{bmatrix} \tag{6.9}$$

Avec ces définitions on construit des matrices MH_0 en posant :

$$MH_0(z) = R_0 W P \prod_{i=1}^{k} D(z_1, z_2) P W R_i W P \tag{6.10}$$

et on considère les filtres H_0 définis par :

$$H_0(z) = H_{0,0}(z_1^2, z_2^2) + H_{0,1}(z_1^2, z_2^2) z_1 + H_{0,2}(z_1^2, z_2^2) z_2 + H_{0,3}(z_1^2, z_2^2) z_1 z_2 \tag{6.11}$$

(les filtres H_1 , H_2 et H_3 sont définis de la même manière)
La synthèse consiste alors à déterminer les angles α_i, β_i pour que la matrice MH_0 satisfasse certaines propriétés ou bien soit optimale par rapport à certains critères.

Le critère que nous avons choisi d'étudier est celui de platitude maximale : les caractéristiques de platitude (directement reliées à la régularité des ondelettes sous-jacentes) se rattachent a deux notions :

- régularité de la fonction limite du processus de subdivision. Pour cela il faut imposer un certain nombre de propriétés sur H_0. En particulier, il est souhaitable qu'un maximum de dérivées partielles du polynôme H_0 s'annulent en $(-1, 1)$, $(-1, -1)$ et $(1, -1)$. Si N_l caractérise l'ordre de platitude du filtre, cela signifie que les équations suivantes sont satisfaites:

$$\forall k_1, k_2,\ k_1 + k_2 \leq N_l, \qquad \frac{\partial^{k_1+k_2} H_0}{\partial z_1^{k_1} \partial z_2^{k_2}}(-1, -1) = 0, \tag{6.12}$$

$$\frac{\partial^{k_1+k_2} H_0}{\partial z_1^{k_1} \partial z_2^{k_2}}(-1, 1) = 0, \tag{6.13}$$

$$\frac{\partial^{k_1+k_2} H_0}{\partial z_1^{k_1} \partial z_2^{k_2}}(1, -1) = 0 \tag{6.14}$$

- Le plus possible de composantes nulles sur les bandes hautes fréquences. Pour cela, il est souhaitable qu'un maximum de dérivées partielles du polynôme H_1, (resp. H_2, H_3) s'annulent aux points $\{(1, 1), (-1, 1)$ et $(1, -1)\}$ (resp. $\{(-1, -1), (-1, 1)$ et $(1, 1)\}$, $\{(-1, -1), (1, 1)$ et $(1, -1)\}$) Comme pour le point précédent, ceci nous fournit alors un ensemble de N_h équations.

Pour fixer les idées sur la démarche à suivre, les équations indexées par N_h engendrent plus de *bonnes* propriétés sur les ondelettes sous-jacentes et sont à considérer en priorité. Par conséquent, le but est de trouver, pour K fixé, la valeur maximale de N_l telle que le système constitué par la première série d'équations admette des solutions réelles. Dans un second temps, c'est N_h que nous ferons varier pour *affiner* les filtres possibles.

6.2.2.2 Application sur un cas connu

Seul le cas $K = 3$ avait été traité dans la littérature (voir [75]) au départ de ce travail, pour la cascade de Vetterli-Kovacevic et une seule famille de filtres avait été exhibée (correspondant aux valeurs $N_l = 2$ et $N_h = 0$. Les figures 6.10 et 6.11 illustrent respectivement la représentation fréquentielle du filtre H_0 obtenu et le filtre itéré limite induit.

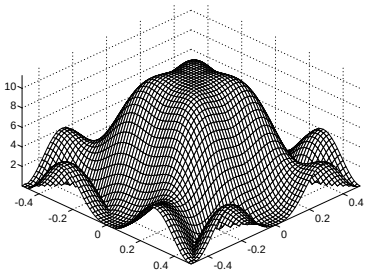


Figure 6.10. Filtre *passe bas* connu pour $K = 3$

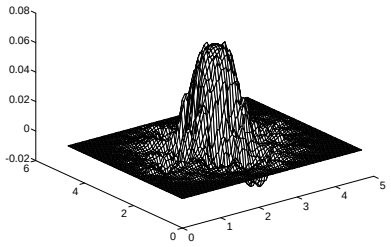


Figure 6.11. Fonction limite (filtre itéré) pour $K = 3$

Calcul d'une valeur limite pour N_l :

Le premier travail a été de vérifier que le problème n'admettait pas de solutions pour $N_l = 2$ et $N_h = 0$. Ceci à été simplement vérifiable en calculant la base de Gröbner du système correspondant. Cette dernière étant égale à [1], le système n'admet donc pas de solutions complexes et a fortiori pas de solutions réelles.

En fixant $N_l = 2$ et en faisant varier N_h jusqu'à trouver un système zéro-dimensionnel ($N_h = 2$), nous avons alors dénombré 64 racines réelles. Par le biais du calcul d'une RUR, nous avons pu déterminer, avec une précision de l'ordre de 10^{-20} l'ensemble des solutions du système et par conséquent, nous avons reconstitué 64 filtres possibles. En fait, le premier polynôme de la RUR se factorise en facteurs de degré 4, ce qui permet d 'exprimer de façon complètement exacte toutes les solutions du système.

En reconstituant alors les polynômes H_0 correspondants, nous obtenons alors 2 solutions distinctes (modulo certaines symétries) dont la solution déjà connue, les caractéristique du nouveau filtre étant données par les figures 6.12 et 6.13.

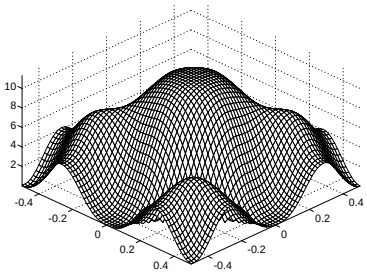


Figure 6.12. Filtre *passe bas* pour $K = 3$

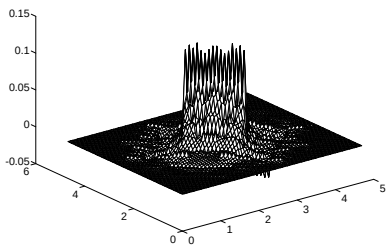


Figure 6.13. Fonction limite (filtre itéré) pour $K = 3$

La découverte de ce nouveau filtre n'apporte pas de progrès en pratique puisque la figure 6.12 tend à montrer que sa représentation fréquentielle est moins *plate* que celle de l'exemple déjà connu et surtout que la fonction limite du banc de filtres itérés est moins régulière. Il est à noter que l'application des algorithmes suggérés dans le chapitre 3 est quasi instantanée sur les systèmes à étudier.

6.2.2.3 Synthèse de nouveaux filtres

Une stratégie similaire à celle utilisée pour l'exemple connu a rapidement permis d'obtenir les valeurs maximales de N_l pour $K = 4, 5, 6, 7, 8$, mais le facteur bloquant a été le calcul des valeurs de N_h : aucune valeur testée n'a permis de retrouver un système zéro-dimensionnel à l'image du cas $K = 3$, l'ajout de contraintes supplémentaires faisant passer d'un système de dimension 1 ou 2 à un système inconsistant. C'est cette application qui, avec le problème d'interpolation de Birkhoff, a motivé l'étude des méthodes de points critiques (calcul d'un point par composante connexe d'une variété algébrique).

Un autre problème de taille est que la formulation directe engendre, pour le cas $K = 8$, $N_l = 6$, $N_h = 6$ par exemple, des polynômes occupant plusieurs dizaines de Méga-octets sur disque (système très redondant).

La reformulation du problème s'appuie sur le remplacement de matrices 4×4 réelles par des matrices 2×2 complexes de forme particulière permettant une identification aisée (positivité de certains déterminants). Précisément, on regarde les produits $P D(z_1, z_2) P$ de la forme :

$$P D(z_1, z_2) P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & z_1 & 0 & 0 \\ 0 & 0 & z_1 z_2 & 0 \\ 0 & 0 & 0 & z_2 \end{bmatrix} \tag{6.15}$$

avec

$$P \begin{bmatrix} 1 \\ z_1 \\ z_2 \\ z_1 z_2 \end{bmatrix} = \begin{bmatrix} 1 \\ z_1 \\ z_1 z_2 \\ z_2 \end{bmatrix} \tag{6.16}$$

et on considère alors

$$H(z_1^2, z_2^2) \begin{bmatrix} 1 \\ z_1 \\ z_2 \\ z_1 z_2 \end{bmatrix} \tag{6.17}$$

En utilisant l'équation 6.16, on forme la cascade à étudier en posant

$$u_0 = \begin{bmatrix} 1 + i\, z_1 \\ (z_1 + i) z_2 \end{bmatrix} \tag{6.18}$$

et en appliquant les règles suivantes :

à chaque multiplication par W, le vecteur courant est multiplié par

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{6.19}$$

à chaque multiplication par R_l, le vecteur courant est multiplié par

$$\begin{bmatrix} e^{i\alpha_l} & 0 \\ 0 & e^{i\beta_l} \end{bmatrix} \tag{6.20}$$

à chaque transformation $P\, D(z_1, z_2) P$, le vecteur courant $u = (u(1), u(2))'$ subit la transformation :

$$u := \begin{bmatrix} \mathcal{R}(u(1)) + i\, z_1 \mathcal{I}(u(1)) \\ z_1 z_2 \mathcal{R}(u(2)) + i\, z_2 \mathcal{I}(u(2)) \end{bmatrix} \tag{6.21}$$

Cette approche a permis de diviser le nombre de variables par 2 et surtout de faire chuter considérablement le nombre de termes dans chaque équation. Les équations finales sont des polynômes en $e^{i\alpha_l}$ et $e^{i\beta_l}$.

Pour réduire le nombre d'équations, nous avons utilisé deux changements de variables intuités par essais successifs. Le premier est :

$$\alpha_1 = a_1 - \frac{a_2 - b_2}{2} \tag{6.22}$$

$$\beta_1 = \frac{a_2 + b_2}{2} \tag{6.23}$$

$$\alpha_k = \frac{a_k - b_k}{2} - \frac{a_{k+1} - b_{k+1}}{2} \text{for } k = 2...K-1 \tag{6.24}$$

$$\beta_k = \frac{a_{k+1} + b_{k+1}}{2} - \frac{a_k + b_k}{2} \text{for } k = 2...K-1 \tag{6.25}$$

$$\alpha_K = \frac{a_K - b_K}{2} \tag{6.26}$$

$$\beta_K = b_1 - \frac{a_K + b_K}{2} \tag{6.27}$$

Le deuxième est issu d'une tentative de calcul incrémental de bases de Gröbner :

$$A_1 = B_1 = \frac{1}{2},\ A_k = e^{i(a_k - a_1)},\ B_k = e^{i(b_k - b_1)} \text{ for } \quad k = 2...K \tag{6.28}$$

Au final on obtient un système très simple modélisant le problème pour tout couple de paramètres (K, N) pour des valeurs suffisamment grandes (vérification par ordinateur des formules) :

Théorème 6.9. ***FR**-[46]. Pour $K \leq 10$,$N \leq 6$, résoudre le problème (K,N) revient à trouver $A_2, ..., A_K$ satisfaisant les équations :*

$$E_2 \qquad \sum_{k=1}^{K} A_k = 0$$

$$E_3 \qquad \sum_{k=2}^{K} \frac{1}{A_k}\left(\sum_{j=1}^{k-1} A_j - \sum_{j=k+1}^{K} A_j \right) = -\frac{1}{4}$$

$$E_4 \qquad \sum_{k=2}^{K-1} \frac{1}{A_k}\left(\sum_{j=k+1}^{K} A_j \right)\left(\sum_{j=1}^{k-1} A_j \right) = -\frac{1}{8}$$

$$E_5 \qquad \sum_{i=4}^{K} \sum_{j=2}^{i-1} \frac{1}{A_i A_j}\left(\sum_{k=1}^{j-1} A_k - \sum_{i+1}^{K} A_k \right)\left(\sum_{k=j+1}^{i-1} A_k \right) = \frac{4K-7}{32}$$

$$E_6 \quad \sum_{k=4}^{K} \frac{1}{A_k} \sum_{i=2}^{k-2} \frac{1}{A_i}\left(\sum_{j=i+1}^{k-1} A_j \right)\left(2\left(\sum_{j=1}^{i-1} A_j \right) + 4\left(\sum_{j=k+1}^{K} A_j \right)\left(\sum_{j=2}^{i-1} A_j \right) \right) = 3$$

(Il faut noter que des valeurs supérieures de K ne sont pas intéressantes d'un point de vue pratique pour des raisons d'efficacité d'implantation)

La différence majeure avec le système écrit de manière directe est que cette fois nous cherchons les solutions complexes de module 1. En remarquant que les équations sont à coefficients rationnels, on peut toutefois utiliser le fait que pour toute solution, son inverse est également solution. On introduit alors de nouvelles variables A'_k et de nouvelles équations $A_k A'_k - 1 = 0$, sachant que les coordonnées A'_k satisfont les mêmes équations que les A_k.

Dans **FR**-[46], l'ensemble des solutions complexes, pour des couples de valeurs (K, N) avec $K = 7, 8$ a alors été obtenu en éliminant les variables A'_k (calcul de base de Gröbner pour un ordre d'élimination) puis en extrayant « à la main » celles de module 1. Depuis, les méthodes de points critiques (voir chapitre 4 ou certaines solutions plus récentes telles que [126]) permettent de décider directement si de tels systèmes admettent ou non des solutions (et de calculer le cas échéant un point par composante connexe de la variété réelle définie par le système), même après un changement de variables pour discriminer les parties imaginaires des parties réelles.

Les systèmes à résoudre étant de dimension positive, un ou 2 degrés de liberté peuvent servir à optimiser les filtres obtenus selon leur utilisation (par exemple pour la compression d'images). Quelques instances ont été proposées dans **FR**-[46], comme par exemple le filtre (8,5) suivant spécialement optimisé pour un critère de symétrie (sa forme paramétrique est l'objet du brevet **FR**-[45]) :

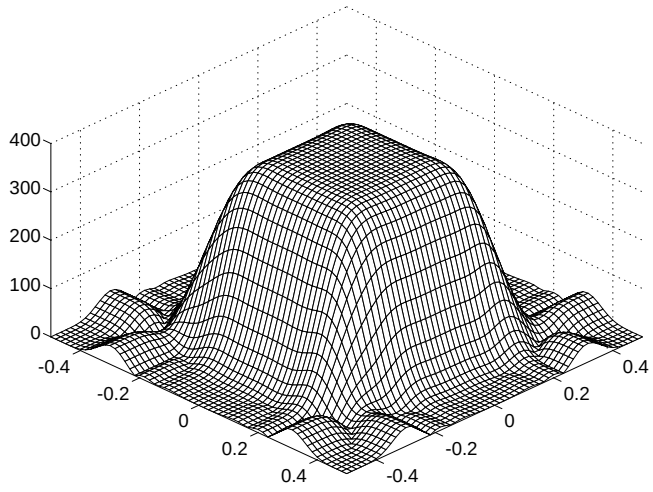


Figure 6.14. Filtre (8,5)

6.3 Systèmes dépendant de paramètres

6.3.1 Classification d'une famille de robots série

Les robots ou manipulateurs étudiés dans cette section sont des robots série à 3 degrés de liberté dont les articulations sont des liaisons pivots telles que les axes de deux laisons successives soient orthogonaux, comme sur la figure suivante :

Ils dépendent de quatre paramètres de conception non nuls notés d_3, d_4, r_2 et r_3 sur le schéma et dans le texte qui suit. Ces manipulateurs ont une structure plus complexe mais offrent de plus grandes possibilités que les robots série couramment utilisés dans l'industrie. En effet, à l'inverse des robots usuels, certains robots de cette famille n'ont pas besoin de passer par une singularité de position (coordonnées (x, y, z) de l'extrémité du manipulateur) pour opérer un changement de posture (changement de configuration articulaire, c'est à dire un autre triplet d'angles $(\theta_1, \theta_2, \theta_3)$ pour une position fixée de l'extrémité du manipulateur).

D'après Wenger et El Omri [145],[143],[102] une condition pour qu'un manipulateur puisse changer de posture sans rencontrer le lieu singulier de son espace articulaire est lié à l'apparition d'un point de rebroussement dans toute section verticale de son espace de travail, d'ou l'adjectif « Cuspidal » utilisé pour caractériser les robots ayant cette propriété.

La caractérisation des robots cuspidaux est donc naturellement algébrique et les roboticiens savent même l'exprimer simplement puisque le problème peut se résumer à tester la présence d'une racine triple dans les zéros d'un polynôme de degré 4 (polynôme caractéristique du manipulateur).

Dans l'étude résumée ici, nous considérons ce polynôme caractéristique dans sa forme la plus générale, c'est à dire en fonction, entre autres, des paramètres de conception d_3, d_4, r_2 et r_3, le paramètre r_1 présent sur la figure pouvant être pris égal à 1.

Le but est de proposer une classification de tous les robots possibles selon leur caractère de cuspidalité en fonction des paramètres de conception.

L'étude de ce type de manipulateur utilise deux espaces :

- L'espace articulaire (ou espace des coordonnées articulaires $(\theta_1, \theta_2, \theta_3)$) peut être vu come le tore de dimension 2 puisque le comportement du manipulateur est invariant par rotation selon l'angle θ_1.

- L'espace opérationnel (ou l'espace des coordonnées cartésiennes $(x,\, y,\, z)$ de l'organe terminal) peut être vu comme un volume de révolution (variation de l'angle θ_1) et on introduira donc $\rho = \sqrt{x^2+y^2}$ pour représenter ses points par des couples (ρ, z).

La relation entre les coordonnées articulaires et les coordonnées cartésiennes est donnée par l'*opérateur géométrique* :

$$\begin{array}{rccc} f\colon & \mathbb{T} & \longrightarrow & \mathbb{R}^+ \times \mathbb{R} \\ & (\theta_2, \theta_3) & \mapsto & (\rho, z) \end{array}$$

avec : $\left\{ \begin{array}{l} \rho^2 = (\sin(\theta_3)\, d_4 + r_2)^2 + (1 + \cos(\theta_2)\, (\cos(\theta_3)\, d_4 + d_3) + \sin(\theta_2)\, r_3)^2 \\ z = \sin(\theta_2)(\cos(\theta_3)\, d_4 + d_3) - \cos(\theta_2)\, r_3 \end{array} \right.$

En posant $c_2 = \cos(\theta_2)$, $c_3 = \cos(\theta_3)$, $s_2 = \sin(\theta_2)$, $s_3 = \sin(\theta_3)$ et en rajoutant les relations $c_2^2 + s_2^2 = 1, c_3^2 + s_3^2 = 1$, on exprime f sous forme algébrique implicite :

$$\left\{ \begin{array}{l} \rho^2 = (s_3\, d_4 + r_2)^2 + (1 + c_2\, (c_3 d_4 + d_3) + s_2 r_3)^2 \\ z = s_2(c_3\, d_4 + d_3) - c_2\, r_3 \\ c_2^2 + s_2^2 = 1 \\ c_3^2 + s_3^2 = 1 \end{array} \right.$$

En éliminant les variables c_2, s_2 par un calcul de base de Gröbner, puis en posant $t = \tan(\frac{\theta_3}{2})$, on retrouve alors le polynôme caractéristique (connu des robotociens) du manipulateur, dont les coefficients dépendent des paramètres de conception ainsi que des variables ρ et z :

$$P(t) = a\, t^4 + b\, t^3 + c\, t^2 + d\, t + e$$

avec :

$$\left\{ \begin{array}{rcl} a & = & m_5 - m_2 + m_0 \\ b & = & -2m_3 + 2m_1 \\ c & = & -2m_5 + 4m_4 + 2m_0 \\ d & = & 2m_3 + 2m_1 \\ e & = & m_5 + m_2 + m_0 \\ m_0 & = & -r^2 + r_2^2 + \frac{(R+1-L)^2}{4} \\ m_1 & = & 2r_2 d_4 + (L - R - 1) d_4 r_2 \\ m_2 & = & (L - R - 1) d_4 d_3 \\ m_3 & = & 2r_2 d_3 d_4^2 \\ m_4 & = & d_4^2 (r_2^2 + 1) \\ m_5 & = & d_4^2 d_3^2 \\ r^2 & = & x^2 + y^2 \\ R & = & r^2 + z^2 \\ L & = & d_4^2 + d_3^2 + r_2^2 \end{array} \right.$$

Les singularités de position sont les points de l'espace articulaire tels que la matrice jacobienne de f ne soit pas de rang plein. En particulier, le complémentaire de ce lieu critique sur le tore est constitué d'au moins 2 composantes connexes.

On note cet ensemble $C_f = \{(\theta_1, \theta_2) \in \mathbb{T},\, \mathrm{Jac}(f) = 0\}$, avec $\mathrm{Jac}(f) = (\cos(\theta_3)\, d_4 + d_3)(\sin(\theta_3 + \cos(\theta_2)\, (\sin(\theta_3)\, d_3 - r_2 \cos(\theta_3)) + \sin(\theta_2)\, r_3\, (\sin(\theta_3)\, d_3 - r_2 \cos(\theta_3))$.

Dans [32], il est démontré que les conditions suivantes sont équivalentes, caractérisant avec précision la notion de manipulateur cuspidal :

Théorème 6.10. *Si on considère un manipulateur générique, c'est à dire pour lequel l'opérateur f est différentiellement stable (tous ses points singuliers sont des singularités stables), alors les propositions suivantes sont équivalentes :*

- *il existe un point de l'espace de travail tel que dans tout voisinage suffisamment petit de ce point, il soit possible de changer de posture sans passer par C_f;*
- *il existe un point de rebroussement de $f(C_f)$ dans l'espace de travail;*

- *il existe un point $p \in C_f$ tel que la tangente à C_f en ce point soit dans le noyau de la différentielle de f en p;*
- *il existe un point dans l'espace de travail en lequel le polynôme P admet une racine triple.*

Etablir une classification des robots série considérés revient donc à discuter l'existence de solutions réelles en fonction de d_3, d_4, r_2, r_3, du système $\{P=0, \frac{\partial P}{\partial t}=0, \frac{\partial^2 P}{\partial t^2}=0, d_4>0, d_3>0, r_2>0, \rho>0\}$, avec $P \in \mathbb{Q}[d_3, d_4, r_2, r_3, z, \rho, t]$. On peut remarquer directement sur les équations que les polynômes sont en ρ^2, si bien qu'il suffira de diviser le nombre de solutions en ρ^2 par 2 pour obtenir le nombre de solutions admissibles de après spécialisation, si bien que le problème se résume à l'étude du système :

$$\mathcal{S} = \{P=0, \frac{\partial P}{\partial t}=0, \frac{\partial^2 P}{\partial t^2}=0, d_4>0, d_3>0, r_2>0\}$$

6.3.1.1 Résolution ad-hoc

L'etude de $\mathcal{S}$ a d'abord été tentée pour $r_3 = 1$, c'est à dire sans décallage pour l'organe terminal. malgré cette simplification ainsi que l'utilisation des variantes les plus érpouvées d'algorithmes d'élimination [44], [50], on a jamais pu calculer directement de bases lexicographiques (avec l'espoir qu'elles fourniraient une paramétrisation rationnelle « générique » des solutions). Pour le moins nous avons toujours arrêté les calculs une fois persuadés que le résultat final serait inexploitable (taille).

Ceci s'explique en partie par la présence de composantes parasites au dessus de valeurs non admissibles des paramètres (par exemple, il y a une composante de dimension 4 au dessus de $d_4 = 0$) issues de cas dégénérés sans intérêt ou simplement d'artéfacts de mise en équations (réduction des dénominateurs, changements de variables). C'est en saturant de multiples fois pour supprimer ou mettre en évidence ces composantes que nous avons finalement obtenu le changement de variables suivant, déterminant pour ce qui suit :

$$\begin{cases} U = & e-a & = & 2(d_4^2+d_3^2+r_2^2-R-1)d_4 r_2 \\ X_1 = & d-b=d-X_3-h\,U & = & 8 r_2 d_3 d_4^2 \\ X_2 = & U+2a-c & = & 4d_4^2(d_3^2-r_2^2-1) \\ h = & r_2/d_3 & & \\ X_3 = & b-h\,U & = & 4 r_2 d_4(1-d_3 d_4) \end{cases}$$

En effet, la base de Gröbner du système pour l'ordre lexicographique $t > a > U > X_1 > X_2 > X_3 > h$ est simple à calculer et donne un système de la forme :

$$G_{\text{lex}=}\begin{cases} \text{surf}_U(U, X_1, X_2, X_3, h) \\ g_{a,1}(a, U, X_1, X_2, X_3, h); \\ g_{a,2}(a, U, X_1, X_2, X_3, h); \\ \vdots \\ g_{a,14}(a, U, X_1, X_2, X_3, h); \\ g_{t,1}(t, a, U, X_1, X_2, X_3, h); \\ g_{t,2}(t, a, U, X_1, X_2, X_3, h); \\ \vdots \\ g_{t,218}(t, a, U, X_1, X_2, X_3, h); \end{cases}$$

La remarque fondamentale pour la suite des calculs est que l'ensemble triangulaire extrait, qui est de la forme :

$$\mathcal{T} = \begin{cases} \text{surf}_U(U, X_1, X_2, X_3, h) \\ g_{a,1}(a, U, X_1, X_2, X_3, h) = \text{lc}_a(X_1, X_2, X_3, h)\, a + c_a(U, X_1, X_2, X_3, h); \\ g_{t,1}(t, a, U, X_1, X_2, X_3, h) = \text{lc}_t(X_1, X_2, X_3, h)\, t + c_t(t, U, X_1, X_2, X_3, h); \end{cases}$$

est régulier est séparable, c'est à dire que les polynômes lc_a et lc_t ne s'annulent sur aucune composante irréductible de $\boldsymbol{V}(\mathcal{T})$. En particulier, ceci qui a pour conséquence que $\langle \mathcal{T} \rangle : (\text{lc}_a \text{lc}_t)^\infty$ est équidimensionnel et radical d'après [6] et que $\mathcal{T}$ donne une paramétrisation rationnelle des points de $\overline{\boldsymbol{V}(\langle G_{\text{lex}} \rangle) \setminus \boldsymbol{V}(\langle \text{lc}_a \text{lc}_t \rangle)}$ avec $\dim(\boldsymbol{V}(\langle G_{\text{lex}} \rangle)) > \dim(\boldsymbol{V}(\langle \text{lc}_a \text{lc}_t c_a c_t \rangle))$.

En revenant aux variables initiales, $\mathcal{T}$ se réécrit :

$$\mathcal{T}' = \begin{cases} \mathrm{surf}(R, d_4, d_3, r_2) = 0 \\ \mathrm{lc}_Z(d_4, d_3, r_2)Z + \mathrm{tr}_Z(R, d_4, d_3, r_2) = 0 \\ \mathrm{lc}_t(d_4, d_3, r_2)t + \mathrm{tr}_t(R, Z, d_4, d_3, r_2) = 0 \end{cases}$$

et donne donc une paramétrisation rationnelle des points de $\overline{\boldsymbol{V}(\langle P, \frac{\partial P}{\partial t}, \frac{\partial^2 P}{\partial t^2}\rangle) \setminus \boldsymbol{V}(\langle \mathrm{lc}_z \mathrm{lc}_t\rangle)}$.

On peut vérifier sans peine, par un calcul d'élimination, que cet ensemble triangulaire est régulier et séparable, si bien que $\dim(\boldsymbol{V}(\langle P, \frac{\partial P}{\partial t}, \frac{\partial^2 P}{\partial t^2}\rangle)) > \dim(\boldsymbol{V}(\langle \mathrm{lc}_z \mathrm{lc}_t\rangle))$ et que l'on a donc ainsi décrit formellement « presque » toutes les points de $\boldsymbol{V}(\langle P, \frac{\partial P}{\partial t}, \frac{\partial^2 P}{\partial t^2}\rangle)$.

Etudier $\overline{\boldsymbol{V}(\langle P, \frac{\partial P}{\partial t}, \frac{\partial^2 P}{\partial t^2}\rangle) \setminus \boldsymbol{V}(\langle \mathrm{lc}_z \mathrm{lc}_t\rangle)}$ à la place de $\boldsymbol{V}(\langle P, \frac{\partial P}{\partial t}, \frac{\partial^2 P}{\partial t^2}\rangle)$ ne nuit pas à la généralité puisqu'il est de toutes façons illusoire de concevoir des robots dont les paramètres de conception vérifient une relation algébrique : toute erreur d'usinage, même infime (et donc inévitable en pratique), de ces pièces nous place immédiatement en dehors de $\boldsymbol{V}(\langle \mathrm{lc}_z \mathrm{lc}_t\rangle)$.

La deuxième remarque, due à la structure de $\mathcal{T}$, est que $P = 0, \frac{\partial P}{\partial t} = 0, \frac{\partial^2 P}{\partial t^2} = 0$ admet une unique solution au dessus de tout ouvert de $\boldsymbol{V}(\mathrm{surf})$ ne rencontrant pas $\boldsymbol{V}(\mathrm{lc}_Z \mathrm{lc}_t)$.

On a donc éliminé les variables Z et t et l'étude du nombre de solutions de $\mathcal{S}$ se résume donc en l'étude du nombre de solutions en R de $\mathcal{S}' = \{\mathrm{surf} = 0, \mathrm{lc}_Z \neq 0, \mathrm{lc}_t \neq 0, d_4 > 0, d_3 > 0, r_2 > 0\}$.

Comme surf est le seul polynôme dépendant de R (lc_t et lc_z ne dépendent que de d_4, d_3 et r_2), on peut étudier séparément $\mathrm{surf} = 0$ et ainsi projeter le problème dans l'espace des paramètres.

Précisément, le nombre de solutions en R de surf $= 0$ ne varie que lorsque l'on rencontre $\mathrm{lc}_R(\mathrm{surf}) = 0$ (coefficient de tête de surf vu comme polynôme en R) ou $\mathrm{discr}_R(\mathrm{surf}) = 0$ (discriminant de surf vu comme polynôme en R). Aussi, le nombre de solutions de $\mathcal{S}$ en (rho, z, t) est nul si $\mathrm{surf} \neq 0$ et sinon ne varie que si on rencontre $\mathcal{D} = \{(d_4, d_3, r_2) \in \mathbb{R}^3, d_4\, d_3\, r_2\, \mathrm{lc}_Z\, \mathrm{lc}_t\, \mathrm{lc}_R\, \mathrm{discr}_R = 0\}$.

Pour conclure, il « suffit » donc de décrire les composantes connexes $\mathcal{C}_1$, ..., $\mathcal{C}_k$ de $\mathbb{R}^3 \setminus \mathcal{D}$, puis de prendre un point u_i dans chacune des $\mathcal{C}_i$ et enfin de résoudre les systèmes $\mathcal{S}_{|(d_4,d_3,r_2)=u_i}$ pour calculer le nombre (constant) de solutions de $\mathcal{S}$ au dessus de chacune des $\mathcal{C}_i$.

En pratique, il suffit donc de :

- (1) calculer une Décomposition Cylindrique Algébrique de $\mathbb{R}^3$ adaptée à $\{d_4, d_3, r_2, \mathrm{lc}_Z, \mathrm{lc}_t, \mathrm{lc}_R, \mathrm{discr}_R\} \subset \mathbb{Q}[d_3, d_4, r_2]$;
- (2) résoudre (compter les solutions réelles) les systèmes zéro-dimensionnels $\{P = 0, \frac{\partial P}{\partial t} = 0, \frac{\partial^2 P}{\partial t^2} = 0\}_{|(d_4,d_3,r_2)=u_i}$, les u_i étant les points tests obtenus pour chaque cellule en (1) en ne considérant que les u_i n'annulant pas $\mathrm{lc}_Z\, \mathrm{lc}_t\, \mathrm{lc}_R\, \mathrm{discr}_R$ et n'ayant aucune coordonnée négative ou nulle (on sait que sinon $\mathcal{S}$ n'admet aucune solution).

En pratique, on ne calcule que les cellules de plus grande dimension d'une CAD de $\mathbb{R}^3$ adaptée à $\{d_4, d_3, r_2, \mathrm{lc}_Z, \mathrm{lc}_t, \mathrm{lc}_R, \mathrm{discr}_R\}$ selon l'algorithme exposé dans le chapitre 5 et on utilise la RUR étendue exposée dans le chapitre 3 pour l'étape (2).

Après simplification des polynômes de $\mathcal{D}$, c'est a dire après factorisation et après avoir enlevé les facteurs n'ayant visiblement aucune racine réelle, on se ramène à calculer une CAD « partielle » de l'ensemble $\mathcal{D} = \{Z_{01} = 0, Z_{02} = 0, f_1 = 0, f_2 = 0, \mathrm{lc}_t = 0\}$ tel que :

- $Z_{01} = d_3^2 r_2^2 + d_3^2 - 2d_3^3 + d_3^4 - d_4^2 + 2d_3 d_4^2 - d_4^2 d_3^2$
- $Z_{02} = d_3^2 r_2^2 + d_3^2 - 2d_3^3 + d_3^4 - d_4^2 - 2d_3 d_4^2 - d_4^2 d_3^2$
- $f_1 = d_3^2 - d_4^2 + r_2^2$
- $f_2 = d_4^2 d_3^6 - d_4^4 d_3^4 + 3d_4^2 d_3^4 r_2^2 - 2d_4^2 d_3^4 + 2d_4^4 d_3^2 - 2r_2^2 d_3^2 d_4^4 + d_4^2 d_3^2 + 3d_4^2 d_3^2 r_2^4 - d_3^2 r_2^2 - 2r_2^2 d_4^4 - d_4^4 r_2^4 - d_4^4 + r_2^6 d_4^2 + d_4^2 r_2^2 + 2r_2^4 d_4^2$
- $\mathrm{lc}_t = -d_3 + d_4 r_2^2 + d_4$

L'union des zéros des polynômes de $\mathcal{D}$ donne la surface suivante :

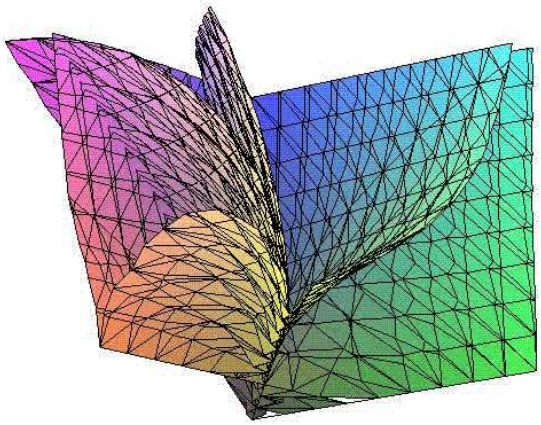


Figure 6.15. Variété discriminante

La première phase de projection (selon d_4) de cet ensemble dans l'algorithme de décomposition cylindrique « partielle » donne les courbes :

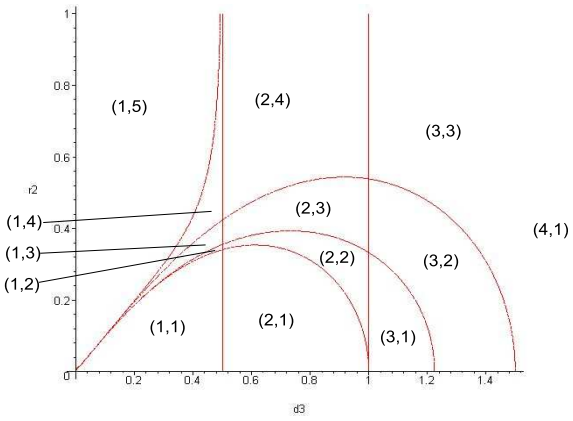


Figure 6.16. CAD d'une variété discriminante

Les cellules du complémentaire des zéros de cette projection ont été numérotées pour un lordre lexicographique sur les coordonnées. Le tableau suivant donne le nombre de solutions du système en fonction pour chaque cellule du complémentaire des zéros de $\mathcal{D}$. Une ligne du tableau représente ce qui se passe au dessus d'une cellule 2 dimensionnelle (première étape de la CAD « partielle »), c'est à dire le nombre de zéros de $\mathcal{S}$ pour des valeurs des paramètres (d_3, r_2) entre deux nappes successives (notées de 0 à 7 et ordonnées selon la coordonnée d_4) de points réels de $\mathcal{D}$:

$(d_3, r_2) \setminus d_4$	1	2	3	4	5	6	7
(1,1)	0	0	4	4	2	0	0
(1,2)	0	4	4	4	2	0	0
(1,3)	0	4	4	4	2	0	0
(1,4)	0	4	4	2	2	0	0
(1,5)	0	4	4	2	0	0	0
(2,1)	0	0	4	4	2	2	0
(2,2)	0	4	4	4	2	2	0
(2,3)	0	4	4	4	2	2	0
(2,4)	0	4	4	2	2	2	0
(3,1)	0	4	4	4	2	2	4
(3,2)	0	4	4	4	2	2	4
(3,3)	0	4	4	2	2	2	4
(4,1)	0	4	4	4	2	2	4

Alors qu'initialement il ne s'agissait que de décider si il existait ou non des points de rebroussement dans une section de l'espace de travail, on obient une information supplémentaire qui est leur nombre. Ceci a permis aux spécialistes d'affiner la classification des robots étudiés selon leur caractère cuspidal en décomposant plus finement l'ensemble des robots cuspidaux [103],[144], etc.

6.3.1.2 Résolution par les variétés discriminantes

Le travail décrit dans la section précédent ne résout le problème que pour $r_3 = 1$, c'est à dire lorsqu'il n'y a aucun décallage de l'organe terminal. La résolution du cas général est nettement plus difficile, et, en particulier, le changement de variable utilisé pour se ramener à l'étude d'un ensemble triangulaire simple n'est plus exploitable en pratique, en tout cas pour des calculs « à la main ». Cette application est à l'origine du travail sur les variétés discriminantes et le cas $r_3 = 1$ peut être résolu très facilement, mais surtout automatiquement, par l'algorithme proposé au chapitre 5.

L'application de cet algorithme montre que $\mathcal{D} = \{Z_{01} = 0, Z_{02} = 0, f_1 = 0, f_2 = 0, \mathrm{lc}_t = 0\}$ est une variété discriminante minimale de $\mathcal{S}$ pour la projection sur l'espace des paramètres, si bien que l'on a pu montrer a postériori que la décomposition obtenue est « optimale ». On peut donc retrouver de façon automatique l'ensemble des résultats présentés dans la section précédente.

Le cas $r_3 \neq 1$ a pu alors être complêtement résolu dans [32]. Les résultats ne sont pas détaillés ici, mais il faut retenir que le calcul nous amène aux limites pratiques de la méthode. En effet, le nombre de cellules générées au final (dans $\mathbb{R}^4$) est de l'ordre de 50 000, ce qui rend la classification finale difficile à exploiter en pratique. Ce calcul a permis toutefois de donner une information qualitative inconnue jusque la puisque l'on déduit des résultats que la proportion de robots cuspidaux est non négligeable (cellules de dimension 4) et que ceux-ci ne peuvent admettre que 0,2,4,6 ou 8 points de rebroussement dans les sections verticales de leur espace de travail.

6.3.2 Certification d'une méthode d'optimisation en traitement du signal

L'application décrite dans cette section traite d'un problème de reconstruction simultanée de signaux provenant de plusieurs sources. On supose que les séquences d'informations transmises sont de la forme

$$x = A\,s + n \tag{6.29}$$

ou x est un vecteur de dimension N (signal observé), A une matrice $N \times K$ (le canal de transmission), s un vecteur de dimension N (signal transmit) et n un vecteur de dimension N (bruit Gaussien). La méthode étudiée (BSS - blind signal separation) consiste à trouver les coefficients d'une matrice W de dimension $N \times K$, dont les lignes seront notées w_i, $i = 1...K$, telle que les vecteurs

$$y_i = w_i^H x, i = 1...K \tag{6.30}$$

représentent les différentes sources du signal, l'exposant ^{H}représentant l'opérateur de transposition conjuguée.

L'algorithme utilisé (CC-CMA - cross-correlation and constant modulus algorithm) minimise la *fonction de coût* suivante pour calculer la séparation des sources :

$$\sum_{i=1}^{K} E[||y_i|^2 - 1|^2] + \frac{\alpha}{2} + \sum_{i=1}^{K} \sum_{j=1, j\neq i}^{K} |E[y_i y_j^\star]|^2 \tag{6.31}$$

ou $\alpha \in \mathbb{R}^{+\star}$ est le paramètre de mélange, $E[.]$ l'espérance et ou * désigne l'opération de conjugaison.

Sous des hypothèses standard pour la méthode étudiée [22] (signaux indépendants à valeurs complexes, présentant certains types de symétries, etc.) et en se limitant à une certaine catégorie de systèmes de transmission (MIMO - [57] and [137]) opérant dans un environnement non bruité, on peut, pour deux sources, modéliser l'ensemble par :

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \tag{6.32}$$

ou g_{ij} est la réponse du i-ème signal provenant de la j-ième source. En posant $k_i = E[|s_i|^4]/E[|s_i|^2]$, on trouve alors que la fonction de coût J s'exprime en fonction de k_i et g_{ij} comme suit :

$$\begin{aligned} J \;=\; & k_1|g_{11}|^4 + k_2|g_{12}|^4 + 4|g_{11}|^2|g_{12}|^2 + k_1|g_{21}|^4 \\ & + k_2|g_{22}|^4 + 4|g_{21}|^2|g_{22}|^2 - 2|g_{11}|^2 - 2|g_{12}|^2 \\ & - 2|g_{21}|^2 - 2|g_{22}|^2 + \alpha(|g_{11}|^2|g_{21}|^2 + |g_{12}|^2|g_{22}|^2 \\ & + g_{11}g_{12}^* g_{21}^* g_{22} + g_{11}^* g_{12} g_{21} g_{22}^*) + 2. \end{aligned} \tag{6.33}$$

En prenant la première dérivée de (6.32) par rapport à g_{11}, g_{12}, g_{21} et g_{22}, on obtient alors :

$$\begin{aligned} \frac{\partial J}{\partial g_{11}} &= g_{11}^*(2k_1|g_{11}|^2 + 4|g_{12}|^2 - 2 + \alpha|g_{21}|^2) + \alpha g_{12}^* g_{21}^* g_{22} \\ \frac{\partial J}{\partial g_{12}} &= g_{12}^*(2k_2|g_{12}|^2 + 4|g_{11}|^2 - 2 + \alpha|g_{22}|^2) + \alpha g_{11}^* g_{21} g_{22}^* \\ \frac{\partial J}{\partial g_{21}} &= g_{21}^*(2k_1|g_{21}|^2 + 4|g_{22}|^2 - 2 + \alpha|g_{11}|^2) + \alpha g_{11}^* g_{12} g_{22}^* \\ \frac{\partial J}{\partial g_{22}} &= g_{22}^*(2k_2|g_{22}|^2 + 4|g_{21}|^2 - 2 + \alpha|g_{12}|^2) + \alpha g_{11} g_{12}^* g_{21}^* \end{aligned} \tag{6.34}$$

Les points stationnaires de l'algorithme CC-CMA utilisé sont tous solutions de ce système. Sans aucune condition additionnelle sur le paramètre α, certains de ces points sont a priori des extrema locaux. La conséquence est alors que les signaux reconstruits sont des combinaisons linéaires des signaux recherchés.

Caractériser la présence de ce type de points stationnaires revient à tester la positivité de la matrice Hessienne « étendue » suivante :

$$H_G J = \begin{pmatrix} \frac{\partial^2 J}{\partial g_{\mathrm{ij}} \partial g_{\mathrm{kl}}^\star} \Big|^2{}_{i,j,k,l=1} & \frac{\partial^2 J}{\partial g_{\mathrm{ij}} \partial g_{\mathrm{kl}}} \Big|^2{}_{i,j,k,l=1} \\ \frac{\partial^2 J}{\partial g_{\mathrm{ij}}^\star \partial g_{\mathrm{kl}}^\star} \Big|^2{}_{i,j,k,l=1} & \frac{\partial^2 J}{\partial g_{\mathrm{ij}}^\star \partial g_{\mathrm{kl}}} \Big|^2_{i,j,k,l=1} \end{pmatrix} \tag{6.35}$$

Précisément, le procédé convergera globalement si cette matrice n'est pas semi-positive.

D'après (6.34) on sait que :

$$\begin{aligned} \frac{\partial^2 J}{\partial g_{11} \partial g_{11}^\star} &= 2k_1|g_{11}|^2 - \alpha \frac{g_{11} g_{12}^\star g_{21}^\star g_{22}}{|g_{11}|^2} \\ \frac{\partial^2 J}{\partial g_{12} \partial g_{12}^\star} &= 2k_2|g_{12}|^2 - \alpha \frac{g_{11}^\star g_{12} g_{21} g_{22}^\star}{|g_{11}|^2}. \end{aligned} \tag{6.36}$$

Comme $\frac{\partial^2 J}{\partial g_{11} \partial g_{11}^\star}$ et $\frac{\partial^2 J}{\partial g_{12} \partial g_{12}^\star}$ doivent être réels, alors :

$$g_{11} g_{12}^\star g_{21}^\star g_{22} = g_{11}^\star g_{12} g_{21} g_{22}^\star = \pm\, |g_{11}||g_{12}||g_{21}||g_{22}|. \tag{6.37}$$

Si $g_{11}g_{12}^{\star}g_{21}^{\star}g_{22} = g_{11}^{\star}g_{12}g_{21}g_{22}^{\star} = |g_{11}||g_{12}||g_{21}||g_{22}|$, la second mineur principal de (6.35) s'écrit :

$$\Delta_2 = 4|g_{11}|^2|g_{12}|^2(k_1k_2-4) - 2\alpha\,|g_{11}||g_{12}||g_{21}||g_{22}|(k_1\frac{|g_{11}|^2}{|g_{12}|^2} + k_2\frac{|g_{12}|^2}{|g_{11}|^2} + 2) \tag{6.38}$$

Dans ce cas, Δ_2 sera négatif si $k_1 < 2$ et $k_2 < 2$. En particulier, $\mathbf{H}_G J$ ne sera pas (semi)positive en un point stationnaire. Ainsi, c'est essentiellement le cas $g_{11}g_{12}^{\star}g_{21}^{\star}g_{22} = g_{11}^{\star}g_{12}g_{21}g_{22}^{\star} = -|g_{11}||g_{12}||g_{21}||g_{22}|$, qui nous intéresse. Les valeurs de α pour lesquelles le procédé sera globalement convergent sont alors solutions du système suivant :

$$\begin{array}{c} |g_{11}|(2k_1|g_{11}|^2 + 4|g_{12}|^2 - 2\alpha\,|g_{21}|^2) - \alpha\,|g_{12}||g_{21}||g_{22}| = 0 \\ |g_{12}|(2k_2|g_{12}|^2 + 4|g_{11}|^2 - 2\alpha\,|g_{22}|^2) - \alpha\,|g_{11}||g_{21}||g_{22}| = 0 \\ |g_{21}|(2k_1|g_{21}|^2 + 4|g_{22}|^2 - 2\alpha\,|g_{11}|^2) - \alpha\,|g_{11}||g_{12}||g_{22}| = 0 \\ |g_{22}|(2k_2|g_{22}|^2 + 4|g_{21}|^2 - 2\alpha\,|g_{12}|^2) - \alpha\,|g_{11}||g_{12}||g_{21}| = 0 \\ 0 < k_1 < 2, 0 < k_1 < 2, \alpha > 0 \\ |g_{11}| > 0, |g_{12}| > 0, |g_{21}| > 0, |g_{22}| > 0 \end{array} \tag{6.39}$$

Le problème se résume donc à trouver des conditions nécessaires et suffisantes sur les paramètres k_1, k_2 et α pour que (6.39) admette des solutions ne rendant pas $H_G J$ semi-positive. En pratique, on peut se contenter d'une version simplifiée en supposant $k_1 = k_2 = 1$. Par exemple, dans le cas de communications par des réseaux sans fil, cela revient à supposer que les utilisateurs partagent les mêmes propriétés statistiques. En supposant $k_1 = k_2 = 1$, on remplace (6.39) par :

$$\begin{array}{c} |g_{11}|(2|g_{11}|^2 + 4|g_{12}|^2 - 2\alpha\,|g_{21}|^2) - \alpha\,|g_{12}||g_{21}||g_{22}| = 0 \\ |g_{12}|(2|g_{12}|^2 + 4|g_{11}|^2 - 2\alpha\,|g_{22}|^2) - \alpha\,|g_{11}||g_{21}||g_{22}| = 0 \\ |g_{21}|(2|g_{21}|^2 + 4|g_{22}|^2 - 2\alpha\,|g_{11}|^2) - \alpha\,|g_{11}||g_{12}||g_{22}| = 0 \\ |g_{22}|(2|g_{22}|^2 + 4|g_{21}|^2 - 2\alpha\,|g_{12}|^2) - \alpha\,|g_{11}||g_{12}||g_{21}| = 0 \\ \alpha > 0, |g_{11}| > 0, |g_{12}| > 0, |g_{21}| > 0, |g_{22}| > 0 \end{array} \tag{6.40}$$

La fin de cette section résume les étapes qui ont permis de montrer le théorème :

Théorème 6.11. ***FR**-[64] La matrice $H_G J$ n'est jamais semi-positive en les zéros de (6.40) sauf pour $\alpha = 1$. Le CC-CMA converge globalement sauf pour $\alpha = 1$.*

La première étape a consisté à calculer une variété discriminante des solutions de (6.40). Pour se mettre en conformité avec les notations du chapitre 5, on pose :

$\mathcal{E} = \{|g_{11}|(2|g_{11}|^2 + 4|g_{12}|^2 - 2\alpha\,|g_{21}|^2) - \alpha\,|g_{12}||g_{21}||g_{22}|,\ |g_{12}|(2|g_{12}|^2 + 4|g_{11}|^2 - 2\alpha\,|g_{22}|^2) - \alpha\,|g_{11}||g_{21}||g_{22}|,\ |g_{21}|(2|g_{21}|^2 + 4|g_{22}|^2 - 2\alpha\,|g_{11}|^2) - \alpha\,|g_{11}||g_{12}||g_{22}|,\ |g_{22}|(2|g_{22}|^2 + 4|g_{21}|^2 - 2\alpha\,|g_{12}|^2) - \alpha\,|g_{11}||g_{12}||g_{21}|)$

$\mathcal{F} = \{\alpha, |g_{11}|, |g_{12}|, |g_{21}|, |g_{22}|\}$

Les indéterminées du problème sont $|g_{11}|, |g_{12}|, |g_{21}|, |g_{22}|$ et l'unique paramètre est α.

En appliquant directement l'algorithme proposé au chapitre 5 pour calculer une variété discriminante de $\mathcal{C} = \{x \in \mathbb{R}^5,\ g(x) = 0,\ f(x) \neq 0,\ g \in \mathcal{E},\ f \in \mathcal{F}\}$, on s'apperçoit que le système est « bien formulé », et que la variété discriminante minimale de $\mathcal{C}$ pour la projection sur l'axe α est l'ensemble des zéros de $P_D = \alpha(\alpha-3)(\alpha-2)(\alpha-1)(\alpha+3)(\alpha+1)$. On obtient alors sans peine le résultat suivant :

Théorème 6.12. ***FR**-[64] Si on définit $\beta = \{|g_{11}| = |g_{12}| = |g_{21}| = |g_{22}| = \frac{1}{\sqrt{3}}\}$, alors*

- *pour $0 < \alpha < 1$ le système (6.40) a 5 solutions incluant (α, β);*
- *pour $\alpha = 1$, le système (6.40) a 3 solutions isolées incluant $(1, \beta)$ et une infinité de solutions vérifiant $|g_{11}| = |g_{21}|, |g_{12}| = |g_{22}|, |g_{21}| = \sqrt{\frac{2 - 3 \star |g_{22}|^2}{3}}, |g_{22}|^2 > 2/3$;*
- *pour $1 < \alpha < \infty$, l'unique solution du système (6.40) est (α, β).*

Pour que $H_G J$ ne soit pas semi-définie positive en un point, il faut et il suffit qu'il existe au moins un de ses mineurs principaux, notés Δ_i par la suite, qui soit négatif.

Pour $1 < \alpha < \infty$, on substituant les indéterminées par les coordonnées de β dans l'expression de $H_G J$ et on remarque alors que Δ_5 est toujours négatif, $H_C J$ ne peut donc pas être semi-définie positive.

Pour $\alpha = 1$, on peut voir facilement que $g_{11} = \frac{\sqrt{2}}{10}$, $g_{12} = \frac{\sqrt{582}}{30}$, $g_{21} = -\, g_{11}$ et $g_{22} = g_{12}$ est solution de $\mathcal{E} = 0$ et satisfait $g_{11} g_{12}^{\star} g_{21}^{\star} g_{22} = g_{11}^{\star} g_{12} g_{21} g_{22}^{\star} = -\frac{97}{7500} < 0$. On peut alors montrer qu'en ce point, $H_G J$ est semi-définie positive puisqu'elle n'admet que des valeurs propres positives ou nulles et qu'elle le reste dans un petit voisinage. Il s'agit donc d'un point stationnaire et l'algorithme CC-CMA ne convergera pas pour $\alpha = 1$.

Par le calcul de variété discriminante, on sait que les solutions de $\mathcal{C} \cap \mathbb{R}^5$ au dessus de $]0, 1[$ forment des courbes continues et ne s'intersectant pas. On montre facilement (calcul d'élimination) que Δ_4 ne s'annule sur aucune de ces branches. Δ_4 est donc de signe constant sur chacune des branches de solutions de $\mathcal{C} \cap \mathbb{R}^5$. En choisissant arbitrairement une valeur $\alpha_0 \in]0, 1[$ et en résolvant le système $\mathcal{C}_{|\alpha=\alpha_0}$ par la méthode décrite au chapitre 3, on montre que Δ_4 est négatif sur toutes les branches sauf la branche $\{\alpha \in]0, 1[, (\alpha, \beta)\}$ ou il est positif.

Il est facile de montrer que Δ_2 est toujours négatif sur cette branche si bien que $H_J G$ admet toujours un mineur principal négatif pour tout $\alpha \in]0, 1[$.

Chapitre 7
Conclusion et perspectives

La structure gigogne des algorithmes présentés dans ce document peut apparaître risquée à terme, mais il faut distinguer les objets calculés des moyens de calcul. Les algorithmes proposés sont, pour la plupart, spécifiés de façon abstraite, ce qui implique qu'il n'y a que très peu de dépendances structurelles.

Par exemple, l'entrée de la RUR n'est pas une base de Gröbner comme le laisseraient penser certaines contributions, mais une description de l'algèbre quotient $\frac{\mathbb{Q}[X_1,..,X_n]}{I}$ (I étant l'idéal induit par les équations du système étudié), c'est à dire une base de $\frac{\mathbb{Q}[X_1,..,X_n]}{I}$ vu comme $\mathbb{Q}$-espace vectoriel de dimension finie et une fonction de projection (forme normale) de $\mathbb{Q}[X_1, .., X_n]$ dans $\frac{\mathbb{Q}[X_1,..,X_n]}{I}$. Les bases de Gröbner fournissent naturellement ce genre de pré-requis, mais d'autres alternatives peuvent être utilisées (par exemple [100] ou [136]).

Il n'empêche que plusieurs paris initiaux ont été effectués sur une *intime conviction*. Le bien-fondé de ces orientations aura pris parfois plusieurs années à être démontré. On peut mentionner, entre autres, le pari *bases de Gröbner* : les attaques en règle à cause d'un unique exemple pour lequel l'objet est connu pour être doublement exponentiel ont fait lâcher prise à bon nombre d'équipes travaillant sur le sujet. Des travaux récents [10],[50],[66],[65] montrent maintenant que l'objet lui-même ainsi que certains algorithmes le calculant tombent dans la bonne classe de complexité pour une large classe de problèmes.

Ce type d'exemples justifie pleinement l'importance accordée aux applications et au développement logiciel puisque cela représente l'unique moyen de se forger une *intime conviction* sur une direction à prendre.

Comme évoqué dans l'introduction de ce document, et, au risque d'être réducteur, il n'existe que deux grandes familles d'algorithmes en géométrie réelle effective couvrant un large spectre de problèmes (on exclut ici les développement spécifiques en optimisation ou sur les courbes et surfaces) :

- les *récursives* : elles reposent essentiellement sur l'algorithme de décomposition algébrique (CAD) de Collins [30] et les outils sous-jacents (résultants, résolutions de polynômes en une variable). L'étude des objets est faite par projections successives, coordonnée par coordonnée. Autant elles fournissent un outil implantable permettant d'aborder les problèmes d'élimination de quantificateurs, autant leur complexité intrinsèquement doublement exponentielle (calcul mais également taille des objets) interdit leur utilisation pour la plupart des problèmes intermédiaires, en tout cas sur les classes d'exemples que nous traitons.

- les *gigognes* : l'idée d'avoir des algorithmes dédiés pour tout une gamme de problèmes intermédiaires n'est pas nouvelle. On retrouve dans [13] un certain nombre d'éléments sur lesquels s'appuie ce projet de recherche (par exemple [11], [14],[12]). En particulier, les preuves d'existence d'algorithmes de bonne complexité pour l'étude des variétés algébriques réelles ont inspiré certains de mes choix. Cependant, la plupart des algorithmes sous-jacents aux études de complexité présentées dans [13] (hormis la résolution des systèmes zéro-dimensionnels qui reprend les algorithmes [Rou99], [RZ03] et [MRR05]) ne sont pas praticables, les constantes de complexité étant a priori très grandes (voir par exemple [Rou03]).

Le travail effectué sur les systèmes zéro-dimensionnels avait pour but initial d'avancer dans la longue liste des algorithmes suggérés par [13], mais la progression sur cette voie prédéfinie s'est arrêtée dès lors qu'il a fallu aborder les problèmes de dimension positive (voir [RRS00], [ARS02]), même si quelques principes généraux (par exemple le calcul de points critiques de fonctions bien choisies) se retrouvent dans certaines contributions (systèmes généraux). Compte tenu du caractère récent de ce type de développement, une difficulté majeure est le manque de références bibliographiques disponibles pour qui veut travailler avec les contraintes qui sont les nôtres (voir introduction de ce document sur les notions d'efficacité et d'algorithmes), expliquant en partie le caractère *self contained* des résultats obtenus et du projet de recherche qui suit.

La réserve d'applications *fil rouge*.

Les applications *fil rouge* sur lesquelles s'appuient les développement du moment et qui fixent les priorités à court et moyen terme sont :

- la synthèse de filtres hyperfréquences (travail commun avec J.-C. Faugère et F. Seyfert dans le cadre de l'Action de Recherche Collaborative INRIA « SILA ») : ce sont des systèmes zéro-dimensionnels de gros degrés (plusieurs milliers) avec un nombre de variables conséquent (jusqu'à 80). Les premiers résultats montrent qu'une fois la RUR calculée, le reste du calcul s'effectue sans trop de problèmes (gros degré mais faible nombre de racines réelles bien réparties).

- l'étude de ridges (voir chapitre 6) : ce sont des systèmes de gros degré (plusieurs milliers de racines complexes) mais avec un faible nombre de variables. La pression est mise sur l'algorithme d'isolation en une variable (la RUR étant, dans ce cas, calculée par un algorithme spécial) sur deux aspects : le tracé rapide de courbes implicites et l'isolation de polynômes de haut degré avec des racines relativement proches. Les exemples significatifs sont actuellement à la limite de ce que l'on peut faire. Nous travaillons à la généralisation du procédé pour des surfaces paramétrées par des fractions rationnelles (et non plus des polynômes).

- la certification de méthodes numériques d'optimisation : les exemples sur lesquels l'effort sera fourni sont des extensions de ceux que l'on peut trouver dans [GLRX06] (passage de 1 à 3 ou 4 paramètres). Ces systèmes ont la particularité d'être de degré élevé (beaucoup de solutions complexes pour des valeurs *génériques* des paramètres) mais de présenter beaucoup de symétries que l'on arrive à exploiter actuellement à la main.

- quelques problèmes ouverts en robotique, particulièrement l'étude des robots parallèles cuspidaux (tirant partie de l'expérience sur les robots cuspidaux série ou encore celle des robots parallèles singuliers (problèmes soumis par Jean-Pierre Merlet). Dans les deux cas, les systèmes à résoudre dépendent de beaucoup de paramètres, mettant la pression sur l'utilisation de la variété discriminante en pratique et surtout de la spécification de la sortie des algorithmes (dépassé 3 ou 4 variables, une décomposition cylindrique de l'espace des paramètres, même partielle, n'est plus exploitable).

Systèmes zéro-dimensionnels.

Systèmes admettant un grand nombre de solutions complexes

Un objectif majeur sur les systèmes zéros dimensionnels est de changer de classe d'exemples. A ce jour, on peut raisonnablement espérer résoudre en des temps décents et sans se poser de questions des systèmes admettant jusqu'à environ un millier de solutions par le biais d'un calcul systématique de RUR. Il faut retenir que la résolution globale de systèmes algébriques est intrinsèquement exponentielle (borne de Bezout); le passage à une classe supérieure de problèmes nécessite donc un effort conséquent.

De l'expérience accumulée, on classe les systèmes en deux familles :

- les systèmes *shape position* (les polynômes définissant les équations engendrent un idéal radical et l'une des coordonnées discrimine les solutions); ils représentent une large part des systèmes provenant d'applications.
- les systèmes non *shape position*; ils sont souvent générés par des processus plus globaux (calcul de variété discriminantes, méthodes de points critiques) ou alors dans certaines familles d'applications (géométrie algorithmique par exemple).

Dans le premier cas, la marge de manoeuvre est faible. La RUR représente a priori un objet incontournable si l'on veut maintenir nos contraintes de certification et son calcul se réduit facilement, pour les systèmes *shape position*, à la résolution d'un simple système linéaire. Mis à part la distribution/parallélisation des calculs (qui se fait naturellement), peu de progrès significatifs sont à attendre. Cependant, les quelques exemples extrêmes que l'on a pu résoudre montrent que l'étape bloquante se situe actuellement dans la dernière phase de calcul : l'approximation certifiée des racines. Les points faibles sont identifiés : le raffinement de racines de polynômes en une variable connaissant un intervalle d'isolation et la substitution de ces intervalles dans la RUR pour obtenir des pavés de taille imposée contenant une unique racine du système.

Dans le deuxième cas, le problème ne se réduit pas à un problème d'algèbre linéaire simple et les classes d'exemples abordables ne sont, de fait, pas les mêmes (degrés inférieurs). Le résultat proposé dans [Rou05] permet de diminuer le temps de calcul de la RUR, connaissant suffisamment d'information sur l'algèbre quotient associée au problème. Il n'en demeure pas moins que le calcul nécessite un changement de variables engendrant une croissance de coefficients non significative en termes de complexité théorique mais rendant plus délicate l'utilisation de la RUR (présence de coefficients de tailles non homogènes). Une idée serait d'utiliser l'algorithme de type LLL [1] pour trouver un changement de variable minimisant ce phénomène, mais on peut émettre des doutes quant au passage à l'échelle de telles stratégies sur les exemples envisagés.

Une autre piste est l'ajout d'une étape intermédiaire permettant de décomposer, ou pour le moins de réduire, l'idéal de polynômes étudié. En effet, dans le cas précis des systèmes qui ne sont pas *shape position*, la décomposition en ensembles triangulaires du système permet de représenter les solutions comme union de solutions de systèmes triangulaires [80, 5, 6] (de la forme $f_1(X_1) = 0$, $f_2(X_1, X_2) = 0$, ..., $f_n(X_1, .., X_n) = 0$) ayant le bon goût d'être des bases de Gröbner dans le cas d'idéaux zéro dimensionnels d'engendrer des idéaux radicaux (ensembles réguliers et séparables). Nous utilisons déjà ce type de décomposition mais nous appliquons systématiquement un calcul de RUR dessus. Ce que nous n'avons pas encore tenté, c'est d'éviter le calcul de RUR et de résoudre directement les systèmes triangulaires. Contrairement à la RUR, la routine de base n'est pas la résolution de polynômes en une variable à coefficients rationnels mais à coefficients réels algébriques, ce qui est a priori nettement plus difficile.

Cependant, si l'on considère un système qui n'est pas *shape position*, les polynômes de la RUR seront de degré le nombre de racines réelles alors que les degrés des polynômes d'une décomposition en ensembles triangulaires seront plus petits. Par exemple, un système admettant 2^n solutions peut être représenté par un ensemble triangulaire où tous les polynômes sont de degré 2 alors que les polynômes d'une RUR seront de degrés 2^n ou 2^{n-1}. Il faut remarquer que les systèmes triangulaires apparaissent de façon naturelle dans certains problèmes (en particulier en géométrie algorithmique) et que leur résolution certifiée est un sujet d'intérêt à lui seul.

Systèmes admettant un faible nombre de solutions complexes

Bien que les activités applicatives récentes forcent à l'étude de gros systèmes, il est d'un intérêt certain de savoir résoudre rapidement des systèmes de taille modeste. La logique est tout à fait différente puisque, par exemple, un programme de résolution *temps réel* interdit structurellement, dans beaucoup de cas, l'utilisation d'arithmétiques multi-précision. Se pose donc la question d'effectuer des calculs utilisant une arithmétique en précision fixe (donc approchée) sans concession sur la qualité de la sortie (exactitude du résultat).

Dans [RZ03], nous montrons que les deux sont conciliables pour l'isolation des racines de polynômes en une variable, mais ceci constitue la seule contribution dans cette voie et ne représente que la phase terminale du processus de résolution. Sur la base des résultats présents dans [Tre02] ainsi que sur quelques résultats expérimentaux de Jean-Charles Faugère, il semble qu'il soit possible de calculer de manière approchée une représentation de l'algèbre quotient associée au problème. Il est connu que l'obtention d'une approximation numérique des solutions peut alors se faire par un calcul de valeurs propres. Les implantations et stratégies existantes ne permettent pas de garantir le résultat et d'après les tests que nous avons effectués, elles ne sont pas concurrentielles des méthodes (semi-numériques) d'homotopie.

Systèmes paramétrés.

Les résultats de [LR06] nous ont permis de pouvoir (enfin) aborder l'étude de systèmes dépendant de paramètres (essentiel des problèmes qui nous sont soumis). Autant nous avons pu montrer la puissance de cette méthode sur des problèmes *génériques* (non sur contraints, zéro-dimensionnels et radicaux pour presque toutes les valeurs des paramètres, polynômes définissant les inégalités ou inéquations ne s'annulant pas sur une composante irréductible de la variété des polynômes définissant les équations), autant beaucoup de points restent à éclaircir pour le cas général. Notons tout de même que les cas *génériques* représentent une large part des systèmes posés.

Comme pour les méthodes de points critiques, les choses se compliquent dès lors que l'on manipule des idéaux non équi-dimensionnels et/ou non radicaux (par exemple, le critère Jacobien ne s'applique plus pour calculer les lieux singuliers et points critiques de projection). Dans le cas du calcul de variétés discriminantes, on peut toutefois, en théorie, faire l'économie d'une décomposition complète (à l'aveugle) des idéaux considérés. Par exemple, isoler les composantes dont la dimension est supérieure à celle de l'espace des paramètres n'est d'aucune utilité. La connaissance de certaines composantes de la variété discriminante (dont les zéros contiennent les projections de points à l'infini), calculables sans conditions sur le système étudié, permet par une simple saturation d'éliminer immédiatement ce type de composantes, faisant gagner un facteur de complexité si l'on adapte légèrement les algorithmes connus (par exemple [52]) de décomposition primaire. On peut d'ailleurs parier que le calcul de ces composantes permet d'améliorer l'algorithme en toute généralité.

Pour l'heure, nous n'avons attaqué que des systèmes admettant un faible nombre de paramètres (3 ou 4 au maximum). L'utilisation de la CAD pour obtenir une décomposition cylindrique algébrique de l'espace des paramètres qui soit compatible avec la variété discriminante ne s'est jamais avérée bloquante (modulo quelques adaptations). Pour rappel, ce procédé nous permet à la fois de décrire les cellules de l'espace des paramètres ne rencontrant pas la variété discriminante et de fournir un point test dans chaque cellule. Les propriétés de la variété discriminante assurant que les solutions sont régulières au dessus de chaque cellule (nombre constant, applicabilité du théorème des fonctions implicites), on sait alors caractériser complètement les solutions du système pour des paramètres n'appartenant pas à la variété discriminante.

Le problème est que la CAD, même modifiée, explosera avec le nombre de paramètres (nombre de cellules et difficulté des calculs). Une alternative est d'utiliser des méthodes de points critiques pour calculer au moins un point dans chaque cellule du complémentaire de la variété discriminante. Notons que ce type de stratégie ne permet pas de décrire les cellules mais simplement de les caractériser, limitant les études aux tests d'existence de solutions ou de calcul du nombre maximal de solutions.

Une autre possibilité est de recoller certaines cellules produites par la CAD afin de limiter la taille de la réponse finale (la CAD produit un nombre doublement exponentiel de cellules alors que le nombre de composantes connexes que l'on désire décrire est intrinsèquement simplement exponentiel). Le problème sous-jacent est celui des cartes routières (déterminer si deux points sont dans la même composante connexe) pour lequel il existe des solutions théoriques effectives [13] mais aucune solution pratique connue. Les travaux récents de Mohab Safey El Din pourraient débloquer la situation [124].

Systèmes généraux

Pour étudier des ensembles constructibles ou semi-algébriques généraux, on peut toujours, dans un premier temps, se ramener à l'étude d'un système paramétré en considérant certaines variables comme des paramètres (par exemple un ensemble maximal de variables transcendantes) et donc envisager de décrire toutes les solutions (décompositions cellulaires, stratifications) d'un système général à l'aide de variétés discriminantes.

Ce type de stratégie se heurte immédiatement à des problèmes de recollement de cellules. En effet, il n'est pas concevable d'obtenir la topologie complète d'un ensemble constructible ou d'un ensemble semi-algébrique en le regardant sous un unique angle : autant le fait de fixer un ensemble de variables transcendantes qui joue le rôle de paramètres permet d'avoir des informations précises sur certaines composantes via la variété discriminante (celles qui se projettent proprement) autant aucune information utile n'est directement calculable pour les composantes ne se projetant pas correctement ou, plus généralement, sur les points sont la projection est contenue dans la variété discriminante.

Bien sûr, on peut réinjecter la variété discriminante dans le système de départ, mais on obtiendra alors, en général, un système paramétré n'ayant aucune solution pour presque toutes les valeurs des paramètres. Il faudra alors, en tout cas dans certains cas, changer l'ensemble de variables transcendantes pour continuer le calcul selon cette stratégie, c'est à dire changer l'angle sous lequel on regarde l'ensemble de points. Ces changements d'angle de vue compliquent terriblement le rassemblement des résultats puisqu'il devient par exemple délicat (pas de représentation cylindriques) de déterminer quelles cellules sont voisines.

Bibliographie

[1] Jr. A. K. Lenstra, H. W. Lenstra and L. Lovàsz. Factoring polynomials with rational coefficients. *Math. Ann.*, 261, 1982.

[2] A.-G. Akritas. There is no "uspensky's method". In *Proceedings of the SYMSAC*, pages 88–90. ACM, New York, 1986.

[3] A. G Akritas, A. Bocharov, and A. Strzebonski. Implementation of real roots isolation algorithms in mathematica. In *Abstracts of the International Conference on Interval and Computer Algebraic Methods in Science and Ingineering*, pages pp. 23–27, 1994.

[4] M.-E Alonso, E. Becker, M.-F. Roy, and T. Wörmann. Multiplicities and idempotents for zerodimensional systems. In *Algorithms in Algebraic Geometry and Applications*, volume 143 of *Progress in Mathematics*, pages 1–20. Birkhäuser, 1996.

[5] P. Aubry. *Ensembles triangulaires de polynômes et résolution de systèmes algébriques*. PhD thesis, Université Paris 6, France, 1999.

[6] P. Aubry, D. Lazard, and M. Moreno-Maza. On the theories of triangular sets. *J. of Symbolic Computation*, 28(1):105–124, 1999. Special Issue on Polynomial Elimination.

[7] P. Aubry, F. Rouillier, and M. Safey. Real solving for positive dimensional systems. *Journal of Symbolic Computation*, 34(6):543–560, 2002.

[8] Auzinger and Stetter. An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations. *Int. Series of Numerical Math.*, 86:11–30, 1998.

[9] B. Bank, M. Giusti, J. Heintz, and G.-M. Mbakop. Polar varieties and efficient real elimination. *Mathematische Zeitschrift*, 238(1):115–144, 2001.

[10] M. Bardet, J.-C. Faugère, and B. Salvy. On the complexity of gröbner basis computation of semiregular overdetermined algebraic equations. In *Proceedings of the International Conference on Polynomial System Solving*, pages 71–74, 2004.

[11] S. Basu, R. Pollack, and M.-F. Roy. On the combinatorial and algebraic complexity of quantifier elimination. *Proc. 35th IEEE Symp. on Foundations of Computer Science*, pages 632–641, 1994.

[12] S. Basu, R. Pollack, and M.-F. Roy. A new algorithm to find a point in every cell defined by a family of polynomials. In *Quantifier elimination and cylindrical algebraic decomposition*. Springer, 1998.

[13] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in real algebraic geometry*, volume 10 of *Algorithms and Computations in Mathematics*. Springer, 2003.

[14] S. Basu, R. Pollack, and M.F. Roy. On the combinatorial and algebraic complexity of quantifier elimination. *Journal of Assoc. Comput. Machin.*, pages 1002–1045, 1996.

[15] E. Becker, T. Mora, M.-G. Marinari, and C. Traverso. The shape of the shape lemma. In *International Symposium on Symbolic and Algebraic Computation*, pages 129–133, 1994.

[16] T. Becker and V. Weispfenning. *Gröbner bases: a computational approach to commutative algebra*. Graduate Texts in Mathematics: readings in mathematics. Springer, 1993.

[17] M. Ben-Or, D. Kozen, and J. Reif. The complexity of elementary algebra and geometry. *Journal of Computation and Systems Sciences*, 32:251–264, 1986.

[18] D.-A. Bini and G. Fiorentino. Mpsolve, version 2.2. `http://www.dm.unipi.it/cluster-pages/mpsolve/index.htm`.

[19] A. Bostan, B. Salvy, and E. Schost. Fast algorithms for zero-dimensional polynomial systems using duality. *Applicable Algebra in Engineering, Communication and Computing*, 14(4):239 – 272, 2003.

[20] R.-P. Brend and H.-T. Kung. Fast algorithms for manipulating formal power series. *J. Assoc.Comput. Mach.*, pages 581–595, 1978.

[21] J.-F. Canny. Some algebraic and geometric computations in pspace. *Twentieth ACM Symp. on Theory of Computing*, pages 460–467, 1988.

[22] L. Castedo, C. J. Escudero, and A. Dapena. A blind signal separation method for multiuser communications. *IEEE Trans. Signal Processing*, 45(5):1343–1348, 1997.

[23] A.-L. Cauchy. *Exercices de Mathematiques Quatrieme Annee*. De Bure Freres, 1929.

[24] F. Cazals, J.-C. Faugère, M. Pouget, and F. Rouillier. The implicit structure of ridges of a smooth parametric surface. Technical Report RR-5608, INRIA, 2005.

[25] F. Cazals, J.-C. Faugère, M. Pouget, and F. Rouillier. Topologically certified approximation of umbilics and ridges on polynomial parametric surfaces. In *Computational Methods for Algebraic Spline Surfaces*, 2006. toappear.

[26] F. Cazals and M. Pouget. Topology driven algorithms for ridge extraction on meshes. Technical Report RR-5526, INRIA, 2005.

[27] G. Collins and A. Akritas. Polynomial real root isolation using descartes' rule of signs. In *SYMSAC*, pages 272–275, 1976.

[28] G. E. Collins and H. Hong. Partial cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 12(3):299–328, 1991.

[29] G.-E. Collins, J. Johnson, and W. Krandick. Interval arithmetic in cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 34(2):145–157, 2002.

[30] G.E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. *Lecture Notes in Computer Science*, 33:515–532, 1975.

[31] G.E. Collins and J.R. Johnson. Quantifier elimination and the sign verification method for real root isolation. In *Proceedings of the ACM-SIGSAM Symposium on Symbolic and Algebraic Computations*, pages 264–271. ACM Press, 1989.

[32] S. Corvez. *Etude de systèmes polynomiaux : contributions à la classification d'une famille de manipulateurs et au calculs des intersections de courbes A-splines*. PhD thesis, Université de Rennes 1, 2005.

[33] S. Corvez and F. Rouillier. Using computer algebra tools to classify serial manipulators. In *Automated Deduction in Geometry*, volume 2930 of *Lecture Notes in Artificial Intelligence*, pages 31–43. Springer, 2003.

[34] D. Cox, J. Little, and D. O'Shea. *Ideals, varieties, and algorithms : an introduction to computational algebraic geometry and commutative algebra*. Undergraduate texts in mathematics. Springer, 1992.

[35] J.-H. Davenport. Computer algebra for cylindrical algebraic decomposition. Technical Report 88-10, University of Bath, School of Math. Sciences, Bath, England, 1985.

[36] F. Budan de Boislaurent. *Nouvelle méthode pour la réésolution des équations numériques d'un degré quelconque*. Paris (1807), 1822.

[37] R. Descartes. La géométrie. DESCARTES, Oeuvres complètes. Publiées par Ch. Adam and P. Tannery, Ed. du Cerf, Paris 1897-1913. Réédition Vrin 1996, vol. VI, pp. 367-485.

[38] P. Dietmaier. The stewart-gough platform of general geometry can have 40 real postures. In J. Lenancic and M. Husty, editors, *Advances in Robot Kinematics: Analysis and Control*, pages 1–10. Kluwer, 1998.

[39] Z. Du, V. Sharma, and C.-K. Yap. Amortized bound for root isolation via sturm sequences. In *Proceedings of the International Workshop on Symbolic-Numeric Computation*, pages 81–93, 2005.

[40] A. Eigenwillig, V. Sharma, and C.-K. Yap. Almost tight recursion tree bounds for the descartes method. In *Proceedings of the International Workshop on Symbolic-Numeric Computation*, page to appear, 2006.

[41] D. Eisenbud, C. Huneke, and W. Vasconcelos. Direct methods for primary decomposition. *Invent. Math.*, 110:207–235, 1992.

[42] A.G. Erdman. *Modern Kinematics*. Wiley, New-York, 1993.

[43] J.-C. Faugère. *Résolution des systèmes d'équations polynômiales*. PhD thesis, Université de Paris VI, 1994.

[44] J.-C. Faugère. A new efficient algorithm for computing gröbner bases (f_4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.

[45] J.-C. Faugère, F. Moreau de Saint-Martin, and F. Rouillier. Une famille de bancs de filtres 2d non séparables. Patent number 97.00528, 1997.

[46] J.-C. Faugère, F. Moreau de Saint Martin, and F. Rouillier. Design of regular nonseparable bidimensional wavelets using gröbner basis techniques. *IEEE SP Transactions Special Issue on Theory and Applications of Filter Banks and Wavelets*, 46(4):845–856, 1998.

[47] J.-C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional gröbner basis by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.

[48] J.-C. Faugère and D. Lazard. The combinatorial classes of parallel manipulators. *Mechanism and Machine Theory*, 30:765–776, 1995.

[49] J.-C. Faugère, J;-P. Merlet, and F. Rouillier. On solving the direct kinematics problem for parallel robots. Technical Report RR-5923, INRIA, 2006. Submitted to J. of Robotics Research.

[50] Jean-Charles Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero f_5. In *Proceedings of the ACM SIGSAM International Symposium on Symbolic and Algebraic Computation*, 2002.

[51] P. Gianni. Properties of gröbner basis under specializations. In *Lecture Notes in Computer Science*, volume 378, pages 293–297. Springer, 1987.

[52] P. Gianni, G. Miller, and B. Trager. Decomposition of algebras. *Lecture Notes in Computer Science*, 356:300–308, 1988.

[53] P. Gianni, B. Trager, and G. Zacharias. Gröbner bases and primary decomposition of polynomial ideals. *Journal of Symbolic Computation*, 6(2):149–168, 1988.

[54] Giusti, Heintz, Morais, and Pardo. When polynomial equation systems can be "solved" fast. In *AAECC-11*, volume 948 of *Lecture Notes in Computer Science*, 1995.

[55] M. Giusti and J. Heintz. Algorithmes - disons rapides - pour la décomposition d'une variété algébrique en composantes irréductibles et équidimensionnelles. In T. Mora and C. Traverso, editors, *Proc. Effective Methods in Algebraic Geometry, MEGA '90*, volume 94 of *Progress in Mathematics*, pages 169–193. Birkhäuser, 1991.

[56] M. Giusti, G. Lecerf, and B. Salvy. A gröbner free alternative for solving polynomial systems. *Journal of Complexity*, 17(1):154–211, 2001.

[57] D. N. Godard. Self-recovering equalization and carrier tracking in two-dimensional data communication systems. *IEEE Trans. Commun.*, 28(11):1867–1875, 1980.

[58] L. Gonzalez-Vega. Applying quantifier elimination to the birkhoff interpolation problem. *Journal of Symbolic Computation*, 22(1):pages 297–310, 1996.

[59] L. González-Vega, H. Lombardi, T. Recio, and M-F. Roy. Sturm-habicht sequence. In *ISSAC-89 Proceedings*, pages 136–146. ACM-Press, 1989.

[60] L. Gonzalez-Vega and I. Necula. Efficient topology determination of implicitly defined algebraic plane curves. *Comput. Aided Geom. Des.*, 19(9):719–743, 2002.

[61] L. González-Vega and G. Trujillo. Using symmetric functions to describe the solution set of a zero dimensional ideal. In G. Cohen, M. Giusti, and T. Mora, editors, *Applied Algebra and Error Correcting Codes*, volume 948 of *Lecture Notes in Computer Science*, pages 232–247. Springer, 1995.

[62] V.E. Gough. Contribution to discussion of papers on research in automobile stability, control and tyre performance, 1956-1957. Proc. Auto Div. Inst. Mech. Eng.

[63] D. Grigoriev and N. Vorobjov. Bounds on numbers of vectors of multiplicities for polynomials which are easy to compute. In *ISSAC: Proceedings of the ACM SIGSAM International Symposium on Symbolic and Algebraic Computation*, 2000.

[64] Nong Gu, D. Lazard, Fabrice Rouillier, and Yang Xiang. Using computer algebra to certify the global convergence of a numerical optimization process. In *Mathematical Aspects of Computer and Information Sciences*, pages 100–112, 7 2006.

[65] A. Hashemi and D. Lazard. Complexity of zero-dimensional gröbner bases. Technical Report RR-5660, INRIA, 2005. submitted to Journal of Symbolic Computation.

[66] A. Hashemi and D. Lazard. Sharper complexity bounds for zero-dimensional gröbner bases and polynomial system solving. Technical Report RR-5491, INRIA, 2005. submitted to Journal of Symbolic Computation.

[67] J. Heintz, M.-F. Roy, and P. Solerno. On the theoritical an practical complexity of the existential theory of reals. *The Computer Journal*, 36(5):427–431, 1993.

[68] H. Hong. Comparison of several decision algorithms for the existential theory of the reals. Technical report, RISC-Linz, Johannes Kepler University, 1991.

[69] K.H. Hunt. *Kinematic geometry of mechanisms*. Clarendon Press, Oxford, 1978.

[70] M.L. Husty. An algorithm for solving the direct kinematic of Stewart-Gough-type platforms. *Mechanism and Machine Theory*, 31(4):365–380, 1996.

[71] C. Innocenti and V. Parenti-Castelli. Direct kinematics in analytical form of a general geometry 5-4 fully parallel manipulator. In P. Kovacs J. Angels and G. Hommel, editors, *Computational kinematics*, pages 141–152. Kluwer, 1993.

[72] J. Gerhardt J. Von Zur Gathen. *Modern Computer Algebra*. Cambridge University Press, 1999.

[73] J.R. Johnson and W. Krandick. Polynomial real roots isolation using approximate arithmetic. In Wolfgang Küchlin, editor, *Proceedings International Symposium On Symbolic And Algebraic Computations*. ACM Press, 1997.

[74] I. Kotsireas and D. Lazard. Central configurations of the 5-body problem with equal masses in three dimensional space. In *Proceedings of CASC*, 1998.

[75] J. Kovacevic and M. Vetterli. "Nonseparable two- and three-dimensional wavelets". *IEEE Transactions on Signal Processing*, 43(5):1269–1272, May 1995.

[76] W. Krandick. Isolierung reeller Nullstellen von Polynomen. In J. Herzberger, editor, *Wissenschaftliches Rechnen*, pages 105–154. Akademie Verlag, Berlin, 1995.

[77] W. Krandick. A data structure for approximation. (research report), 1996.

[78] W. Krandick. Trees and jumps and real roots. *Journal of Computational and Applied Mathematics*, 162(1):51–55, 2003.

[79] J.-M. Lane and R.-F. Riesenfeld. Bounds on a polynomial. *BIT*, 21:112–117, 1981.

[80] D. Lazard. Solving zero - dimensional algebraic systems. *Journal of Symbolic Computation*, 13:117–132, 1992.

[81] D. Lazard. Stewart platforms and gröbner basis. In *Proceedings of Advances in Robotics Kinematics*, pages 136–142, 1992.

[82] D. Lazard. On the representation of rigid-body motions and its application to generalized platform manipulators. In J. Angeles and al., editors, *Computational Kinematics*, volume 28 of *Solid Mechanics and its Applications*, pages 175–182. Kluwer, 1993.

[83] D. Lazard. On the specification for solvers of polynomial systems. In *5th Asian Symposium on Computers Mathematics - ASCM 2001*, volume 9 of *Lecture Notes Series in Computing*, pages 66–75. World Scientific, 2001.

[84] D. Lazard. Central configurations of four gravitational masses with an axys of symmetry. Workshop on Applications of Commutative Algebra - Catania 2002, Catania, Italie, 2002.

[85] D. Lazard. Injectivity of real rational mappings: the case of a mixture of two Gaussian laws. *Math. Comput. Simulation*, 67(1-2):67–84, 2004.

[86] D. Lazard and F. Rouillier. Solving parametric polynomial systems. *Journal of Symbolic Computation*, 2006. to appear.

[87] G. Lecerf. *Une alternative aux méthodes de réécriture pour la résolution des systèmes algébriques. PhD thesis*. PhD thesis, École polytechnique, France, 2001.

[88] T. Maekawa, F. Wolter, and N. Patrikalakis. Umbilics and lines of curvature for shape interrogation. *Computer Aided Geometric Design*, 13:133–161, 1996.

[89] K. Mahler. An inequality for the discriminant of a polynomial. *Michigan Mathematical Journal*, 11:257–262, 1964.

[90] M.G. Marinari, H.M. Möller, and T. Mora. On multiplicities in polynomial system solving. *Transactions of the American Mathematical Society*, 348(8):283–321, 1996.

[91] J. P. Merlet. *Les robots parallèles*. Hermès, 1990.

[92] J-P. Merlet. Solving the forward kinematics of a Gough-type parallel manipulator with interval analysis. *Int. J. of Robotics Research*, 23(3):221–236, 2004.

[93] G. Moroz. Complexity of the resolution of parametric systems of equations and inequations. In *proceedings of the ISSAC'06 conference*, pages 246–253, 2006. Best student paper award.

[94] R. Morris. *Symmetry of Curves and the Geometry of Surfaces: two Explorations with the aid of Computer Graphics*. PhD thesis, University of Liverpool, 1990.

[95] B. Mourrain. The 40 generic positions of a parallel robot. In *ISSAC'93*, pages 173–182, 1993.

[96] B. Mourrain. About the rational map associated to a parallel robot. Technical Report 2141, INRIA, Sophia-Antipolis, 1993.

[97] B. Mourrain. An introduction to linear algebra methods for solving polynomial equations, 1998.

[98] B. Mourrain, F. Rouillier, and M.-F. Roy. Bernstein's basis and real root isolation. In *Combinatorial and Computational Geometry*, volume 52 of *MSRI Publications*, pages 459–478. Cambridge University Press, 2005.

[99] B. Mourrain, M. Vrahatis, and J.-C. Yakoubsohn. On the complexity of isolating real roots and computing with certainty the topological degree. *Journal of Complexity*, 18(2):612–640, 2002.

[100] Bernard Mourrain and Philippe Trébuchet. Generalized normal forms and polynomial system solving. In M. Krauers, editor, *International Symposium on Symbolic and Algebraic Computation*, pages 253–260. ACM Press, 2005. Distinguished paper award.

[101] M. Noro and K. Yokoyama. A modular method to compute the rational univariate representation of zero-dimensional ideals. *Journal of Symbolic Computation*, 28(1-2):243–263, 1999.

[102] J. El Omri. *Analyse géomtrique et cinématique des mécanismes de type manipulateur*. PhD thesis, Université de Nantes, 1996.

[103] M. Baili P. Wenger, D. Chablat. A dh parameter based condition for 3r orthogonal manipulators to have four distinct inverse kinematic solutions. *ASME Journal of Mechanical design*, 127(1):150–155, 2005.

[104] P. Pedersen. *Counting Real Zeros*. PhD thesis, New York University, 1991.

[105] P. Pedersen, M.-F. Roy, and A. Szpirglas. Counting real zeros in the multivariate case. In *Computational Algebraic Geometry*, volume 109 of *Progress in Mathematics*, pages 61–76. Birkhäuser, 1993.

[106] M. Raghavan. The Stewart platform of general geometry has 40 configurations. In *ASME Design and Automation Conf.*, volume 32-2, pages 397–402, Chicago, 1991.

[107] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. *Journal of Symbolic Computation*, 13:255–352, 1992.

[108] N. Revol and F. Rouillier. A library for arbitrary precision interval arithmetic. In *10th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics (scan2002)*, 2002.

[109] N. Revol and F. Rouillier. Motivations for an arbitrary precision interval arithmetic and the mpfi library. In R. Baker Kearfott, editor, *Workshop on Validated Computing, Toronto - Canada*, pages 155–161. SIAM, 2002.

[110] N. Revol and F. Rouillier. Motivations for an arbitrary precision interval arithmetic and the mpfi library. *Reliable Computing*, 11:1–16, 2005.

[111] B. Roth. Computation in kinematics. In P. Kovacs J. Angeles, G. Hommel, editor, *Computational Kinematics*, pages 3–14. Kluwer, 1993.

[112] F. Rouillier. Real roots counting for some robotics problems. In *Computational Kinematics*, pages 73–82. Kluwer, 1995.

[113] F. Rouillier. *Algorithmes efficaces pour l'étude des zéros réels des systèmes polynomiaux*. PhD thesis, Université de Rennes I, 1996.

[114] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Journal of Applicable Algebra in Engineering, Communication and Computing*, 9(5):433–461, 1999.

[115] F. Rouillier. Efficient agorithms based on the critical point method. In *DIMACS Workshop on Algorithmic and Quantitative Aspects of Real Algebraic Geometry in Mathematics and Computer Science*, 2001.

[116] F. Rouillier. Efficient algorithms based on critical points method. In *Algorithmic and Quantitative Real Algebraic Geometry*, volume 60 of *Discrete Mathematics and Theoretical Computer Science*, pages 123–138. American Mathematical Society, 2003.

[117] F. Rouillier. On solving zero dimensional systems with rational coefficients. Submitted to Journal of Symbolic Computation, 2005.

[118] F. Rouillier, M.F. Roy, and M. Safey. Finding at least one point in each connected component of a real algebraic set defined by a single equation. *Journal of Complexity*, 16(4):716–750, 2000.

[119] F. Rouillier, M. Safey, and E. Schost. Solving the birkhoff interpolation problem via the critical point method: An experimental study. In J. Richter-Gebert and D. Wang, editors, *Automated Deduction in Geometry*, number 2061 in Lecture Notes in Artificial Intelligence, pages 26–40. Springer, 2001.

[120] F. Rouillier and P. Zimmermann. Efficient isolation of a polynomial real roots. Technical Report RR-4113, INRIA, 2001.

[121] F. Rouillier and P. Zimmermann. Efficient isolation of polynomial real roots. *Journal of Computational and Applied Mathematics*, 162(1):33–50, 2003.

[122] M.F. Roy and A. Szpirglas. Complexity of the computation on real algebraic numbers. *Journal of Symbolic Computation*, 10(1):39–52, 1990.

[123] M. Safey El Din. *Résolution Réelle des Systèmes Polynomiaux en Dimension Positive*. PhD thesis, Université de Paris VI, 2001.

[124] M. Safey El Din. Generalized critical values and testing sign conditions on a polynomial. In D. Wang and Z. Zheng, editors, *Proceedings of International Conference on Mathematical Aspects of Computer and Information Sciences*, pages 61–84, 2006.

[125] M. Safey El Din and E. Schost. Polar varieties and computation of one point in each connected component of a smooth real algebraic set. In J.R. Sendra, editor, *International Symposium on Symbolic and Algebraic Computation 2003 - ISSAC'2003, Philadelphie, USA*, pages 224–231. ACM Press, aug 2003.

[126] M. Safey El Din and E. Schost. Properness defects of projection functions and computation of at least one point in each connected component of a real algebraic set. *Journal of Discrete and Computational Geometry*, 32(3):417–430, sep 2004.

[127] M. Safey El Din and P. Trébuchet. Strong bihomogeneous bézout theorem and degree bounds for algebraic optimization. *Journal of Pure and Applied Algebra*, under revision.

[128] E. Schost. *Sur la résolution des systèmes polynomiaux à paramètres*. PhD thesis, École polytechnique, 2000.

[129] A. Seidenberg. A new decision method for elementary algebra. Annals of Mathematics, 60:365–374,, 1954.

[130] A. Seidl and Th. Sturm. A generic projection operator for partial cylindrical algebraic decomposition. Technical Report MIP-0301, Fakultät für Mathematik und Informatik, Universität Passau, 2003.

[131] V. Shoup. Efficient computation of minimal polynomials in algebraic extensions of finite fields. In *ISSAC*, 1999.

[132] A.J. Sommese, J. Verschelde, and C.W. Wampler. Advances in polynomial continuation for solving problems in kinematics. *ASME J. of Mechanical Design*, 126(2):262–268, 2004.

[133] H. Sterk, H. Cuypers, and A. M. Cohen, editors. *Some Tapas of Computer Algebra*. Algorithms and Computation in Mathematics. Springer, 1998.

[134] C. Sturm. Mémoire sur la résolution des équations numériques. *Inst. France Sc. Math. Phys. 6*, 1835.

[135] B. Sturmfels. *Solving Systems of Polynomial Equations*. Number 97 in CBMS Regional Conferences Series. Amer.Math.Soc., 2002.

[136] P. Trébuchet. *Vers une résolution stable et rapide des équations algébriques*. PhD thesis, Université Pierre et Marie Curie, 2002.

[137] J. R. Treichler and B. G. Agee. A new approach to multipath correction of constant modulus signals. *IEEE Trans. Signal Processing*, 31(2):459–472, 1983.

[138] J.V. Uspensky. *Theory of equations*. McGraw-Hill Book Company, 1948.

[139] A.J.-H. Vincent. Sur la résolution des equations numériques. *Journal de Mathématiques Pures et Appliquees*, pages 341–372, 1836.

[140] C.W. Wampler. Forward displacement analysis of general six-in-parallel SPS (Stewart) platform manipulators using soma coordinates. *Mechanism and Machine Theory*, 31(3):331–337, 1996.

[141] C.W. Wampler, A.P. Morgan, and A.J. Sommese. Numerical continuation methods for solving polynomial systems arising in kinematics. *ASME J. of Mechanical Design*, 112:59–68, 1990.

[142] D. Wang. *Elimination Methods*. Springer, 2001.

[143] P. Wenger. Classification of 3r positioning manipulators. *Journal of Mechanical Design*, 120:327–332, June 1998.

[144] P. Wenger. Uniqueness domains and regions of feasible paths for cuspidal manipulators. *IEEE Transactions on Robotics*, 20(4):745–750, 2004.

[145] Ph. Wenger and J. El Omri. Changing posture for cuspidal robot manipulators. In *Proceeding of the 1996 IEEE Int. Conf on Robotics and Automation*, pages 3173–3178, 1996.

Résumé

Cette présentation résume un ensemble de méthodes générales pour la résolution des systèmes d'équations (et d'inéquations/inégalités) polynomiales, axées principalement sur l'étude de leurs racines réelles ; Il s'agit essentiellement de « résoudre » des systèmes admettant ou non une infinité de solutions, dépendant ou non de paramètres. Le sujet étant vaste, on le contraint en imposant que les algorithmes soient exacts ou certifiés, ou, en d'autres termes, que les résultats ne présentent aucune ambigüité du point de vue de l'utilisateur (nombre ou structure des solutions, approximations numériques lorsque cela a un sens, caractère réel des solutions, etc.).

Le travail exposé a été effectué avec deux objectifs principaux : progresser dans la mise au point de méthodes effectives utiles en géométrie réelle, mais surtout proposer quelques alternatives crédibles aux outils standards de calcul scientifique, avec pour objectif de répondre à quelques problèmes ouverts (ou réputés difficiles) dans divers domaines applicatifs.

Ce bilan de plusieurs années de travail est, en particulier, l'occasion de montrer les efforts spécifiques a priori nécessaires pour arriver à quelques résultats probants (implantation efficace, coût de la certification, balance entre les efficacités théoriques et pratiques) et s'articule autour de quelques applications « fil rouge » ayant motivé les principaux choix théoriques, algorithmiques voir techniques.

Abstract

This presentation summarizes a set of general methods for solving systems of polynomial equations (with or without inequalities), mainly centered on the study of their real roots; the main subject is the "resolution" of systems with an arbitrary number of solutions (finite or infinite), depending or not on parameters. Since the field is vast, one constrains it by imposing it that the algorithms are exact or certified or, in other words, that the results are never ambiguous from the user's point of view (number or structure of the solutions, numerical approximations when it makes sense, real character of the solutions, etc.).

The exposed work was carried out with two principal objectives: to progress in the development of useful effective methods in real geometry, but especially to propose some credible alternatives to the standard tools for scientific computations, with the objective of answering to some open problems (or considered difficult) in various applications.

This assessment of several years of work is, in particular, the occasion to show the specific efforts which seem to be necessary to obtain convincing results (efficient implementations, cost of the certification/exactness, balance between theoretical and practical efficiency, etc.) and is articulated around some selected applications which justify the main theoretical, algorithmic but also technical choices.