Introduction à l'arithmétique par intervalles

Nathalie Revol

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Introduction à l'arithmétique par intervalles

Nathalie Revol, Projet Arénaire, LIP, École Normale Supérieure de Lyon et laboratoire ANO,

Université des Sciences et Technologies de Lille

No 4297

Octobre 2001

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--4297--FR

RHÔNE-ALPES

Introduction à l'arithmétique par intervalles

Nathalie Revol, Projet Arénaire, LIP, École Normale Supérieure de Lyon et laboratoire ANO, Université des Sciences et Technologies de Lille

Thème 2 — Génie logiciel
et calcul symbolique
Projet Arénaire

Rapport de recherche n° 4297 — Octobre 2001 — 33 pages

Résumé : Cet article est une introduction à l'arithmétique par intervalles. Avec une telle arithmétique, il est possible à la fois de tenir compte des incertitudes sur les données et de retourner un encadrement contenant à coup sûr le résultat d'un calcul : la force de l'arithmétique par intervalles est en effet la fiabilité des résultats.

L'objectif de cette introduction est de mettre en évidence les points forts d'une telle arithmétique et de montrer comment contourner ses faiblesses. Son avantage majeur est de fournir une information globale telle qu'un surencadrement de l'image d'un ensemble par une fonction. Cette information globale peut être utilisée pour déterminer le caractère contractant d'une itération et par conséquent pour prouver l'existence et l'unicité de la solution calculée. Elle peut aussi servir à optimiser globalement une fonction en évitant de se laisser piéger par un optimum local.

Mots-clé : arithmétique par intervalles, fiabilité, effet enveloppant, dépendance des données, itération de Gauss-Seidel par intervalles, méthode de Newton par intervalles, algorithme de Hansen pour l'optimisation globale.

(Abstract: pto)

Ce texte est également disponible sous forme de rapport de recherche du Laboratoire de l'Informatique du Parallélisme, cf. `http://www.ens-lyon.fr/LIP`.

Unit´e de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN (France)
T´el´ephone : 04 76 61 52 00 - International: +33 4 76 61 52 00
T´el´ecopie : 04 76 61 52 52 - International: +33 4 76 61 52 52

Introduction to interval arithmetic

Abstract: This paper constitutes an introduction to interval arithmetic. This arithmetic allows on the one hand to take into account the measurement uncertainties on data and on the other hand to determine an enclosure of the computed result that is guaranteed to contain it: indeed, the main advantage of interval arithmetic is its reliability.

The goal of this introduction is to emphasize the strong points of interval arithmetic and to explain how to alleviate its problems. The main advantage is to provide global information, such as for instance the range of a function over a whole set. This global information can serve to prove that an iteration is contractant and thus that it has a fixed point. It can also be used to detemine the global optimum of a function without being trapped by a local one.

Key-words: interval arithmetic, reliable computation, wrapping effect, data dependency, Rump's algorithm for linear systems, interval Newton method for nonlinear systems, Hansen's algorithm for global optimization.

1 Motivation pour changer d'arithmétique

L'accroissement actuel de la puissance des ordinateurs, qu'il s'agisse des processeurs seuls ou de leur utilisation en parallèle, a entraîné une augmentation du volume des calculs numériques. Les exigences sur ces calculs ne portent plus seulement sur la quantité des calculs effectués, mais également sur la qualité : le besoin d'arithmétiques fiables se fait sentir de façon marquée.

Quelques problèmes numériques ont été assez spectaculaires pour devenir connus du grand public et mettre en évidence ce besoin. Citons un accident survenu pendant la guerre du Golfe : le 25 février 1991, un anti-missile Patriot du système de défense anti-aérien basé à Dhahran en Arabie Saoudite a échoué dans sa mission de poursuite et d'interception d'un Scud. Ce Scud a donc pu atteindre des bâtiments militaires et a tué 28 Américains. Ce dysfonctionnement du système de défense a été analysé comme suit : *The Patriot battery at Dhahran failed to track and intercept the Scud missile because of a software problem in the system's weapons control computer. This problem led to an inaccurate tracking calculation that became worse the longer the system operated. At the time of the incident, the battery had been operating continuously for over 100 hours. By then, the inaccuracy was serious enough to cause the system to look in the wrong place for the incoming Scud.* (cf. `http://www.fas.org/spp/starwars/gao/im92026.htm`).

Un autre exemple où les erreurs de calculs ont conduit à une erreur notable (et heureusement moins dramatique) est le cas de l'indice de la Bourse de Vancouver (d'après la page Web de Pete Stewart [69]). En 1982, la Bourse de Vancouver a créé un nouvel indice avec une valeur nominale de 1 000. Après chaque transaction boursière, cet indice était recalculé et tronqué après le troisième chiffre décimal et, au bout de 22 mois, la valeur obtenue était 524,881, alors que la valeur correcte était 1 098, 811. Cette différence s'explique par le fait que toutes les erreurs d'arrondi étaient dans le même sens : l'opération de troncature diminuait à chaque fois la valeur de l'indice.

Ces deux exemples peuvent paraître anecdotiques et peu significatifs des dangers de l'arithmétique flottante usuelle. Or il s'avère que même les limites théoriques de validité numérique de certains algorithmes sont actuellement dépassées. Le calcul de la factorisation QR d'une matrice rectangulaire permet de résoudre des problèmes linéaires au sens des moindres carrés ; ce calcul est connu pour être en pratique numériquement très stable. De façon théorique, cette stabilité a été établie (cf. [30]), pour les matrices carrées pour simplifier, sous l'hypothèse que $n^3u < 1/2$ où n désigne la dimension de la matrice et u représente la différence entre 1^+, le premier nombre flottant (c'est-à-dire représentable sur ordinateur en utilisant l'arithmétique flottante) strictement supérieur à 1 et $1 : u = 1^+ - 1$. En arithmétique IEEE double précision (qui est la plus courante actuellement — pour une présentation de l'arithmétique flottante et de la norme IEEE, voir [35, 26, 21]), u vaut 2^{-53} et donc n doit être inférieur à $\sqrt[3]{1/2u} \simeq 2.10^5 = 200\,000$. Or il existe des simulations numériques faisant intervenir des résolutions de systèmes linéaires de dimension $20\,000\,000 = 2.10^7$: par exemple pour étudier l'écoulement de l'air autour d'un avion [77], un maillage de 4 millions de points est utilisé et en chacun de ces points sont déterminées les 3 composantes de la vitesse, la pression et la densité, ce qui correspond bien à 20 millions de variables. Notons au passage qu'un tel calcul requiert l'utilisation d'ordinateurs parallèles... et que les algorithmes parallèles sont souvent de moins bonne qualité numérique que leurs homologues séquentiels [23, 12] ! On peut toujours se rassurer avec la remarque que, les matrices étant creuses, le nombre d'opérations est très inférieur à n^3 et la borne de stabilité n'est peut-être pas encore atteinte. De plus, l'étude théorique de la stabilité numérique est souvent pessimiste puisqu'il s'agit de majorations des pires erreurs, comparée au comportement pratique moyen des algorithmes...

L'utilisation de l'arithmétique flottante est motivée par le fait que les calculs sont très peu coûteux puisque la représentation des nombres est de longueur fixe, ce qui permet l'implantation des opérations sur ces nombres dans le matériel. Cependant nous venons d'illustrer qu'ils ne sont ni précis ni garantis.

D'autres arithmétiques ont été proposées pour obvier à ces problèmes. Un premier type d'arithmétique précise est fort répandu, il s'agit de l'arithmétique exacte qui manipule essentiellement des entiers et des rationnels [76]. Elle est disponible via les systèmes de calcul formel tels que Maple, largement utilisé dans les cursus de premier cycle (classes préparatoires en particulier) et Mathematica pour ne citer que les plus célèbres (mais pas gratuits) ou via des bibliothèques, dont la plus connue doit être GMP (GNU Multiple Precision arithmetic library), développée par Granlund et disponible à `http://www.swox.com/gmp`. Dans cette arithmétique, tout nombre est représenté exactement, c'est-à-dire sans erreur, et tout calcul fournit également un résultat exact, y compris dans des corps finis ou dans des extensions algébriques de $\mathbb{Q}$, ce qui

ne permettent *a priori* pas les autres arithmétiques mentionnées ici. Cependant, un problème bien connu est l'explosion de la taille des nombres qui interviennent dans les calculs (par taille on entend le nombre de chiffres de leur écriture ou de façon équivalente la mémoire requise pour les stocker). Cette explosion entraîne un coût élevé de calcul, puisque chaque opération a un coût (en nombre d'opérations élémentaires telles que des opérations booléennes sur les bits ou en nombre d'instructions assembleur sur des registres de longueur fixe) au moins linéaire en la taille de ses opérandes. Une autre limitation de l'arithmétique exacte est que certaines fonctions ne sont pas disponibles : il s'agit des fonctions qui font sortir du domaine des nombres représentables, qui est souvent $\mathbb{Q}$ ou quelques extensions algébriques de $\mathbb{Q}$, telles que la racine carrée, l'exponentielle ou les fonctions trigonométriques, même s'il est possible de les manipuler.

Un autre type d'arithmétique, conçue pour imiter les fonctionnalités de l'arithmétique flottante tout en offrant une précision accrue, est l'arithmétique multi-précision. Tout comme en arithmétique flottante, les nombres représentés sont des approximations des réels et toutes les fonctions usuelles ($\sqrt{}$, log, sin par exemple) sont disponibles. Cependant la taille de ces nombres n'est pas limitée à la longueur du mot machine mais est laissée au choix de l'utilisateur, autrement dit la précision est plus grande qu'en arithmétique flottante, d'où le vocable « multi-précision ». Dans une telle arithmétique, les opérations ne peuvent plus être réalisées directement par le matériel (puisque la taille des nombres ne correspond plus à la longueur d'un registre) et sont de complexité, en termes d'opérations élémentaires, qui est fonction de la taille des nombres. Cependant cette taille de nombre reste la même pour les opérandes et pour le résultat d'une opération, à la différence du calcul exact où le résultat d'une multiplication par exemple a une taille grossièrement égale à la somme des tailles des multiplicandes ; cette caractéristique implique que le coût d'une opération (en mémoire et en temps) est moindre en arithmétique multi-précision qu'en arithmétique exacte [21, 79]. Néanmoins, même si la perte de précision est retardée, cette arithmétique souffre des mêmes défauts que l'arithmétique flottante, à savoir l'absence de garantie sur le résultat ou même d'informations sur les erreurs commises.

L'arithmétique que nous allons présenter dans cet article est une arithmétique qui offre des résultats garantis : il s'agit de l'arithmétique par intervalles [52, 21, 2, 78], pour laquelle tout nombre est représenté par un intervalle le contenant, dont les bornes sont représentables dans l'arithmétique sous-jacente, et tout calcul fournit un intervalle qui encadre le résultat cherché. Comme la plupart des implantations de cette arithmétique sont basées sur l'arithmétique flottante, les opérations sont peu coûteuses. Cependant certaines propriétés mathématiques ne sont plus vérifiées dans une telle arithmétique, ce qui nécessite une grande vigilance de l'utilisateur lors de l'écriture des algorithmes. De plus, il est souvent difficile d'échapper au grossissement des intervalles, autrement dit à une faible précision des résultats. Nous allons développer ces différents aspects dans la suite de cet article.

La prochaine partie sera consacrée à la définition de l'arithmétique par intervalles et à la mise en évidence de certaines propriétés (ou pertes de propriétés), ainsi qu'au problème de l'évaluation d'une fonction f définie sur $\mathbb{R}$ pour un argument intervalle. Dans la troisième partie, nous montrerons que pour tirer parti de cette arithmétique, il est nécessaire de revoir les algorithmes numériques usuels mais qu'il est possible de résoudre certains problèmes pour lesquels les autres arithmétiques citées ne fournissent que des solutions partielles. En effet, remplacer simplement le type « float » ou « double » des variables par le type « interval » dans un algorithme d'élimination de Gauss conduit souvent à des résultats inexploitables (typiquement $]-\infty;+\infty[$ parce qu'un pivot contient 0) et il faut veiller à utiliser des algorithmes contractants pour éviter de tels résultats. Nous présenterons l'algorithme de Rump pour la résolution de systèmes linéaires, l'algorithme de Newton pour la résolution de systèmes non linéaires et l'algorithme de Hansen pour l'optimisation d'une fonction continue. Ce dernier est sans équivalent en algorithmique numérique usuelle. Dans la dernière partie, nous présenterons brièvement quelques logiciels mettant à disposition cette arithmétique ou l'utilisant.

2 Arithmétique par intervalles

2.1 Définitions et opérations

Le principe fondamental de l'arithmétique par intervalles consiste à remplacer tout nombre par un intervalle le contenant et à effectuer les calculs sur des intervalles, tout intervalle calculé contenant le résultat du calcul exact.

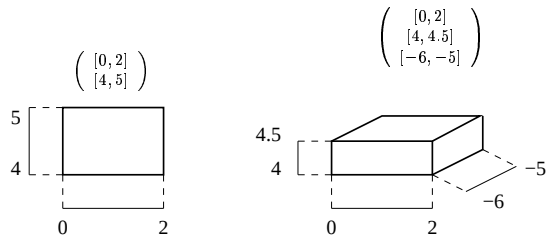

FIG. 1 – *Exemples de vecteurs de $I\!R^2$ et $I\!R^3$.*

Le premier intérêt est de pouvoir prendre en compte les incertitudes de mesure dans le cas où les données sont des valeurs expérimentales. Le second intérêt est de permettre de manipuler sur ordinateur, en utilisant le calcul flottant, des quantités qui ne sont pas exactement représentables : par exemple, le nombre π sera représenté par $[3,14;,3,15]$ sur une machine qui calcule en base 10 avec 3 chiffres de mantisse. Cet intervalle contient de façon *garantie* la valeur de π et tout calcul impliquant cet intervalle contiendra le résultat du calcul avec π. Cette propriété de résultats garantis constitue l'avantage essentiel de l'arithmétique par intervalles ; comme il est de plus possible de préserver cette propriété avec une implantation basée sur l'arithmétique flottante, son intérêt essentiel provient de son effectivité. En effet, et ce même si les données sont représentables exactement, les calculs intermédiaires sont arrondis et une bonne implantation de l'arithmétique par intervalles prend en compte ces erreurs d'arrondi. Tout nombre est donc remplacé par un intervalle le contenant et représentable sur ordinateur : par exemple il est représenté par ses extrémités qui sont des nombres flottants, ou par son milieu et son rayon qui sont également des nombres flottants. Dans un premier temps, les notions essentielles de l'arithmétique par intervalles seront introduites sans se soucier d'une réalisation effective sur ordinateur, c'est-à-dire que toute opération sera considérée comme exacte ; nous reviendrons sur l'adaptation des résultats énoncés au cas de l'arithmétique flottante au §2.3.

Dans ce qui suit, un intervalle sera représenté par ses extrémités : $\boldsymbol{x} = [\underline{x}; \bar{x}] = \{x \in I\!R \,/\, \underline{x} \leq x \leq \bar{x}\}$. Tout nombre réel x sera confondu avec l'intervalle $[x;x]$ correspondant. On notera par $I\!I\!R$ l'ensemble des intervalles fermés bornés sur $I\!R$. Les quantités intervalles seront désormais repérées par des caractères gras et les caractères italiques non gras désigneront les quantités ponctuelles. On peut alors définir un vecteur d'intervalles dans $I\!I\!R^n$ comme un n-uplet d'intervalles (cf. figure 1) et une matrice d'intervalles dans $\mathcal{M}_{m\times n}(I\!I\!R)$ comme une matrice de taille $m \times n$ dont les composantes sont des intervalles, comme dans l'exemple de la matrice 3×3 ci-dessous :

$$\begin{pmatrix} [0;2] & [1;2] & [-5;-4] \\ [-1;1] & [2;3] & [1;5] \\ [-2;0] & [-2;1] & [-2;-1] \end{pmatrix}$$

On désignera souvent par les termes « pavé » ou « boîte » un élément de $I\!I\!R^n$.

On notera par $\mathrm{mid}(\boldsymbol{x})$ ou parfois $\check{x}$ le milieu de l'intervalle $\boldsymbol{x} = [\underline{x}; \bar{x}]$: $\mathrm{mid}(\boldsymbol{x}) = \frac{\underline{x}+\bar{x}}{2}$ et par $w(\boldsymbol{x})$ la largeur (w pour *width*) de $\boldsymbol{x}$: $w(\boldsymbol{x}) = \bar{x} - \underline{x}$. Le rayon de $\boldsymbol{x}$, $\mathrm{rad}(\boldsymbol{x})$, est la moitié de la largeur de $\boldsymbol{x}$. La *mignitude* d'un intervalle est le minimum des valeurs absolues : $\mathrm{mig}(\boldsymbol{x}) = \min\{|x|, x \in \boldsymbol{x}\}$. Si $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots \boldsymbol{x}_n)$ est un pavé de $I\!R^n$, $\mathrm{mid}(\boldsymbol{x})$ est le vecteur des milieux des composantes $\boldsymbol{x}_i$ de $\boldsymbol{x}$ et $w(\boldsymbol{x})$ (resp. $\mathrm{rad}(\boldsymbol{x})$) est le maximum des largeurs (resp. rayons) des $\boldsymbol{x}_i$.

Il est possible d'étendre les opérations arithmétiques usuelles aux intervalles grâce à la formule suivante : si $\boldsymbol{x}$ et $\boldsymbol{y}$ sont deux intervalles et $\diamond \in \{+, -, *, /\}$ alors (si $\diamond$ est l'opération de division, $\boldsymbol{y}$ ne doit pas contenir 0)

$$\boldsymbol{x} \diamond \boldsymbol{y} = \{x \diamond y \,/\, x \in \boldsymbol{x},\ y \in \boldsymbol{y}\}.$$

En pratique, on peut calculer ce résultat en n'utilisant que les extrémités des intervalles $\boldsymbol{x}$ et $\boldsymbol{y}$:

$$
\begin{array}{lcll}
[\underline{x};\bar{x}] & + & [\underline{y};\bar{y}] & = [\underline{x}+\underline{y};\bar{x}+\bar{y}], \\
[\underline{x};\bar{x}] & - & [\underline{y};\bar{y}] & = [\underline{x}-\bar{y};\bar{x}-\underline{y}], \\
[\underline{x};\bar{x}] & * & [\underline{y};\bar{y}] & = [\min(\underline{x}*\underline{y},\underline{x}*\bar{y},\bar{x}*\underline{y},\bar{x}*\bar{y});\max(\underline{x}*\underline{y},\underline{x}*\bar{y},\bar{x}*\underline{y},\bar{x}*\bar{y})], \\
[\underline{x};\bar{x}]^2 & & & = \left\{ \begin{array}{ll} [\bar{x}^2;\underline{x}^2], & \text{si } \forall x \in \boldsymbol{x}, x \leq 0, \\ [0;\max(\underline{x}^2,\bar{x}^2)], & \text{si } 0 \in \boldsymbol{x}, \\ [\underline{x}^2;\bar{x}^2] & \text{si } \forall x \in \boldsymbol{x}, 0 \leq x, \end{array} \right. \\
[\underline{x};\bar{x}] & / & [\underline{y};\bar{y}] & = [\min(\underline{x}/\underline{y},\underline{x}/\bar{y},\bar{x}/\underline{y},\bar{x}/\bar{y}); \\
 & & & \quad \max(\underline{x}/\underline{y},\underline{x}/\bar{y},\bar{x}/\underline{y},\bar{x}/\bar{y})] \text{ avec } 0 \notin \boldsymbol{y}.
\end{array}
$$

On peut également étendre les opérations algébriques telles que la racine carrée et les fonctions élémentaires au cas d'opérandes intervalles :

- si $\boldsymbol{x} = [\underline{x};\bar{x}] \geq 0$, c'est-à-dire que $\forall x \in \boldsymbol{x}$, $x \geq 0$ alors $\sqrt{\boldsymbol{x}} = [\sqrt{\underline{x}};\sqrt{\bar{x}}]$;
- si $\boldsymbol{x} = [\underline{x};\bar{x}] > 0$ alors $\ln \boldsymbol{x} = [\ln \underline{x};\ln \bar{x}]$;
- si $\boldsymbol{x} = [\underline{x};\bar{x}]$ alors $\exp \boldsymbol{x} = [\exp \underline{x};\exp \bar{x}]$;

on profite ici de la monotonie des fonctions calculées.

Dans le cas où la fonction calculée n'est pas monotone, les formules sont un peu plus complexes :

$$
|[\underline{x};\bar{x}]| = \left\{ \begin{array}{ll} [\underline{x};\bar{x}] & \text{si } \forall x \in \boldsymbol{x}, x \geq 0, \\ [-\bar{x};-\underline{x}] & \text{si } \forall x \in \boldsymbol{x}, x \leq 0, \\ [0;\max(-\underline{x},\bar{x})] & \text{si } 0 \in \boldsymbol{x}, \end{array} \right.
$$

et par exemple

$$
\begin{array}{rcl}
 & \sin[\pi/3;\pi] & = [0;1] \\
\text{et} & \sin[0;10] & = [-1;1].
\end{array}
$$

Une première remarque est que le résultat d'une opération arithmétique est plus large que les opérandes : par exemple $w(\boldsymbol{x} \pm \boldsymbol{y}) = w((\boldsymbol{x}) + w(\boldsymbol{y})$. Il est à noter également que les opérations arithmétiques sur les intervalles ne vérifient pas les propriétés algébriques de leurs analogues scalaires. En particulier la soustraction n'est pas la réciproque de l'addition et la division n'est pas la réciproque de la multiplication : $[-2;3]+[5;7] = [3;10]$ mais le résultat de cette opération moins le second opérande vaut $[3;10]-[5;7] = [-4;5]$ et n'est pas égal à $[-2;3]$, il le contient.

Il existe des définitions de l'arithmétique par intervalles qui visent à préserver ces propriétés, citons en particulier les travaux de Markov [46] : les intervalles peuvent avoir leurs bornes « à l'envers », c'est-à-dire qu'un intervalle $\boldsymbol{x} = [\underline{x};\bar{x}]$ peut vérifier $\underline{x} > \bar{x}$; un tel intervalle peut être considéré comme un « anti-intervalle » puisque le soustraire de l'intervalle « à l'endroit » $[\bar{x};\underline{x}]$, en utilisant la formule donnée pour la soustraction, retourne 0 : $[\bar{x};\underline{x}]-[\underline{x};\bar{x}] = [\bar{x}-\bar{x};\underline{x}-\underline{x}] = [0;0]$. Dans cette présentation nous ne considérerons pas ce cas car un intervalle « à l'envers » en fin de calcul nous paraît être un résultat difficile à interpréter...

Une autre propriété algébrique « perdue » lors du passage aux intervalles est la distributivité de la multiplication par rapport à l'addition. En arithmétique par intervalles on a seulement sous-distributivité :

$$
\forall \boldsymbol{x},\boldsymbol{y},\boldsymbol{z}, \quad \boldsymbol{x}*(\boldsymbol{y}+\boldsymbol{z}) \subset \boldsymbol{x}*\boldsymbol{y}+\boldsymbol{x}*\boldsymbol{z},
$$

par exemple si $\boldsymbol{x} = [-1;2]$, $\boldsymbol{y} = [-4;-3]$ et $\boldsymbol{z} = [5;7]$,

$$
\boldsymbol{x}*(\boldsymbol{y}+\boldsymbol{z}) = [-4;8] \subset \boldsymbol{x}*\boldsymbol{y}+\boldsymbol{x}*\boldsymbol{z} = [-15;18]
$$

et l'inclusion est stricte dans ce cas.

Le triplet $(I\!R,+,0)$ est donc un monoïde commutatif, ainsi que $(I\!R,*,1)$.

Cette propriété de sous-distributivité implique que les ensembles $I\!R^n$ ne sont pas des espaces vectoriels sur $I\!R$. Cela n'interdit cependant pas de définir sur ces ensembles les opérations analogues aux opérations sur $I\!R^n$ et le calcul matriciel à l'aide des matrices d'intervalles.

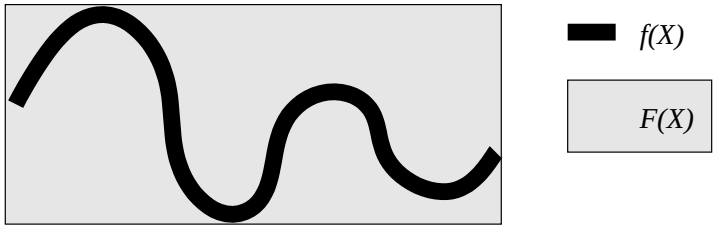

FIG. 2 – *Effet enveloppant : le plus petit pavé contenant l'image (en noir) de la fonction f est le pavé grisé.*

2.2 Fonctions par intervalles

À partir des opérations et fonctions définies sur des intervalles au paragraphe précédent, on peut étendre pour les intervalles toute fonction, dès qu'elle est définie par une expression ne faisant intervenir que ces calculs. Soit f une fonction de $D \subset \mathbb{R}^m$ dans $\mathbb{R}^n$, la propriété fondamentale de garantie des résultats permet d'assurer que pour tout pavé $\boldsymbol{x} \subset D$, le pavé obtenu en substituant les variables scalaires par les composantes intervalles correspondantes de $\boldsymbol{x}$ est un surencadrement de l'image de $\boldsymbol{x}$ par f. L'idéal serait de déterminer le plus petit pavé contenant $f(\boldsymbol{x})\ldots$

Commençons par examiner le cas des fonctions polynomiales à une seule variable. Si $f : \mathbb{R} \to \mathbb{R}$ est définie par $x \mapsto x^2 - 2x + 1$ et si $\boldsymbol{x} = [-1;3]$, en remplaçant x par $\boldsymbol{x}$ dans l'expression de f, on obtient $\boldsymbol{x}^2 - 2\boldsymbol{x} + 1 = [-5;12]$. Si on utilise plutôt l'expression — équivalente en arithmétique réelle — $f(x) = x(x-2)+1$, on obtient $\boldsymbol{x}(\boldsymbol{x}-2)+1 = [-8;4]$ et enfin en utilisant l'écriture factorisée $f(x) = (x-1)^2$ on obtient $(\boldsymbol{x}-1)^2 = [0;4] = f(\boldsymbol{x})$. Cet exemple illustre clairement le fait que des expressions équivalentes en arithmétique réelle ne le sont plus en arithmétique par intervalles, même si chacune donne lieu à un surencadrement de l'image de $\boldsymbol{x}$ par f.

Pour une fonction quelconque, l'évaluation optimale de cette fonction sur un intervalle est un but inaccessible : en effet, le problème *a priori* plus simple de l'évaluation à ε près d'une fonction polynomiale à plusieurs variables et à coefficients rationnels est NP-dur [24].

2.2.1 Problèmes de l'arithmétique par intervalles : dépendance des données et effet enveloppant

Nous venons de mettre le doigt sur l'inconvénient majeur de l'arithmétique par intervalles, à savoir la surestimation (ou encadrement trop large) des résultats.

Deux phénomènes ont été identifiés comme explicatifs de ce problème. Le premier problème, connu sous le nom d'« effet enveloppant » ou *wrapping effect*, est illustré par la figure 2 : si f est une fonction de $\mathbb{R}^m$ dans $\mathbb{R}^n$, l'image par f d'un pavé $\boldsymbol{x}$ de $\mathbb{R}^m$ sera un ensemble de $\mathbb{R}^n$ de forme quelconque. Or l'arithmétique par intervalles impose que le résultat calculé soit un pavé de côtés parallèles aux axes qui contient $f(\boldsymbol{x})$ et ce pavé peut être de volume beaucoup plus grand que celui de $f(\boldsymbol{x})$. Une solution pour remédier à ce problème consiste à découper $\boldsymbol{x}$ en sous-pavés et à prendre pour encadrement de $f(\boldsymbol{x})$ la réunion des images de ces sous-pavés. Le résultat n'est alors plus un pavé mais peut être très proche de l'image exacte, cf. §2.2.2.

Le second phénomène expliquant la surestimation des résultats a été observé dans les exemples précédents, il s'agit du problème de dépendance des données (*data dependency*) ou décorrélation des données : lors du remplacement du calcul de $x * x$ par $\boldsymbol{x} * \boldsymbol{x}$, le résultat est $\{x * y \,/\, x \in \boldsymbol{x},\, y \in \boldsymbol{x}\}$, autrement dit l'égalité entre x et y est perdue. De même, dans l'énoncé de la sous-distributivité, l'écriture $\boldsymbol{x} * \boldsymbol{y} + \boldsymbol{x} * \boldsymbol{z}$ est la traduction de $\{x * y + x' * z \,/\, x \in \boldsymbol{x}, x' \in \boldsymbol{x}, y \in \boldsymbol{y}, z \in \boldsymbol{z}\}$ et ne tient pas compte de l'identité de x et x', c'est pour cette raison que cet intervalle est plus large que $\boldsymbol{x} * (\boldsymbol{y} + \boldsymbol{z})$ dans lequel la variable $\boldsymbol{x}$ n'apparaît qu'une seule fois et ne peut donc pas être décorrélée de ses autres occurrences.

Cette remarque est à la base du théorème suivant, dû à Moore [51].

Théorème. Si f est une fonction continue de $\mathbb{R}^n$ dans $\mathbb{R}$ définie par une expression dans laquelle chaque variable x_i apparaît au plus une fois, alors pour tout pavé $\boldsymbol{x} \subset \mathbb{R}^n$, l'évaluation par intervalles obtenue en remplaçant x_i par la composante correspondante de $\boldsymbol{x}$ est exactement égale à $f(\boldsymbol{x})$.

Ce problème de dépendance et plus précisément la propriété de sous-distributivité de la multiplication par rapport à l'addition permet de démontrer le théorème suivant [4, 21]. Ce théorème fournit une stratégie d'évaluation des polynômes pour lesquels on ne connaît pas de forme factorisée ni de forme « optimale » au sens du théorème de Moore.

Théorème. Parmi toutes les évaluations par intervalles d'une fonction polynomiale donnée sous forme développée, l'évaluation à l'aide du schéma de Hörner est celle qui donne le meilleur résultat, autrement dit le plus petit résultat pour l'inclusion.

2.2.2 Évaluation des fonctions : fonctions d'inclusion naturelles

Il apparaît désormais clairement que l'objectif à atteindre en calcul par intervalles n'est pas le plus petit intervalle au sens de l'inclusion, mais simplement un surencadrement modérément large.

Si on désire évaluer l'image d'une fonction à variables scalaires sur un vecteur d'intervalles, on définit une extension intervalle $\boldsymbol{f}$ de f à partir d'une expression définissant f. Cette fonction $\boldsymbol{f}$ est appelée *fonction d'inclusion naturelle* de f. En particulier, cette extension $\boldsymbol{f}$ vérifie que pour tout intervalle ponctuel x (avec l'abus de notation $x = \{x\}$ pour les points) $\boldsymbol{f}(x) = f(x)$. Une fonction d'inclusion naturelle est également monotone pour l'inclusion, c'est-à-dire que

$$\forall \boldsymbol{x}, \forall \boldsymbol{y}, \text{ si } \boldsymbol{x} \subset \boldsymbol{y} \text{ alors } \boldsymbol{f}(\boldsymbol{x}) \subset \boldsymbol{f}(\boldsymbol{y}).$$

Cette propriété semble aller de soi, nous verrons au §2.2.3 que certaines méthodes d'évaluation ne la vérifient pas. On peut en effet définir une *fonction d'inclusion* ou *extension intervalle* $\boldsymbol{f}$ d'une fonction scalaire $f : D \subset \mathbb{R}^n \to \mathbb{R}$ par la simple propriété d'inclusion

$$\forall \boldsymbol{x} \in \mathbb{IR}^n, \ \boldsymbol{x} \subset D, \ f(\boldsymbol{x}) \subset \boldsymbol{f}(\boldsymbol{x}).$$

La propriété de monotonie par l'inclusion a déjà été mentionnée, deux autres caractéristiques intéressantes sont

- le fait d'être « fine » : une fonction d'inclusion $\boldsymbol{f}$ de f est fine si pour tout vecteur $x \in \mathbb{R}^n$, on a $\boldsymbol{f}(x) = f(x)$;
- le fait d'être convergente : une fonction d'inclusion $\boldsymbol{f}$ de f est convergente si pour toute suite de pavés $(\boldsymbol{x}_k)_{k\in\mathbb{N}}$,
$$\lim_{k\to+\infty} w(\boldsymbol{x}_k) = 0 \Rightarrow \lim_{k\to+\infty} w(\boldsymbol{f}(\boldsymbol{x}_k)) = 0.$$

Si f est une fonction à valeurs dans $\mathbb{R}^n$, l'étude d'une fonction d'inclusion pour f revient à l'étude d'une fonction d'inclusion pour chacune des composantes de f ; c'est pour cette raison que nous nous concentrerons sur l'étude des fonctions à valeurs réelles.

Afin de pouvoir comparer la qualité de deux évaluations par intervalles, une distance est nécessaire. On peut munir $\mathbb{IR}$ d'une structure d'espace métrique grâce à la distance (de Hausdorff) q définie par

$$\begin{aligned} q(\boldsymbol{x}, \boldsymbol{y}) &= \min\{q \in \mathbb{R}^+ \ / \ \boldsymbol{x} \subset \boldsymbol{y} + [-q; q] \text{ et } \boldsymbol{y} \subset \boldsymbol{x} + [-q; q]\} \\ &= \max(|\underline{\boldsymbol{x}} - \underline{\boldsymbol{y}}|, |\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}}|) \\ &= |\check{\boldsymbol{x}} - \check{\boldsymbol{y}}| + 1/2|w(\boldsymbol{x}) - w(\boldsymbol{y})|. \end{aligned}$$

Comme très souvent $\boldsymbol{x}$ sera l'intervalle minimal et $\boldsymbol{y}$ sera une surestimation de $\boldsymbol{x}$: $\boldsymbol{y} \supset \boldsymbol{x}$, cette distance $q(\boldsymbol{x}, \boldsymbol{y}) = \max(|\underline{\boldsymbol{x}} - \underline{\boldsymbol{y}}|, |\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}}|)$ mesurera de combien $\boldsymbol{y}$ « déborde » de chaque côté de $\boldsymbol{x}$.
La distance classiquement utilisée dans $\mathbb{IR}^n$ est le maximum des distances pour chacune des composantes :

$$\text{si } \boldsymbol{x} = (\boldsymbol{x}_1, \ldots \boldsymbol{x}_n) \in \mathbb{IR}^n, \ \boldsymbol{y} = (\boldsymbol{y}_1, \ldots \boldsymbol{y}_n) \in \mathbb{IR}^n, \ q(\boldsymbol{x}, \boldsymbol{y}) = \max_{1\le i\le n} q(\boldsymbol{x}_i, \boldsymbol{y}_i).$$

Les différents exemples du paragraphe précédent mettent en évidence le fait que l'objet à étudier est l'expression et non la fonction qu'elle représente. Nous allons adopter la démarche de Neumaier [53] et donner uniquement des résultats sur des expressions, qui sont informellement des programmes sans test n'utilisant que les fonctions arithmétiques et algébriques et les fonctions élémentaires usuelles. L'avantage

de baser l'étude sur l'expression arithmétique utilisée plutôt que sur la fonction scalaire correspondante est mis en évidence dans l'exemple suivant : $\sqrt{\boldsymbol{x} - \boldsymbol{x}}$ correspond à la fonction scalaire $f : x \mapsto \sqrt{x - x} = 0$ alors que cette expression n'est pas définie pour tout intervalle non réduit à un point, en effet pour tout intervalle $\boldsymbol{x}$, la différence $\boldsymbol{x} - \boldsymbol{x} = [-w(\boldsymbol{x}); w(\boldsymbol{x})]$ contient des réels strictement négatifs si $w(\boldsymbol{x}) > 0$.

Définition : expression arithmétique. Une expression arithmétique en les variables $\xi_1, \ldots \xi_n$ est définie par :

i) toute constante est une expression ;

ii) ξ_i, $1 \leq i \leq n$ est une expression ;

iii) si g et h sont des expressions et si $\diamond \in \{+, -, *, /, \hat{}\}$ alors $g \diamond h$ est une expression ;

iv) si g est une expression et si φ est une fonction élémentaire, $\varphi \in \{|.|, ^2, \sqrt{}, \exp, \ln, \sin, \cos, \arctan\}$ alors $\varphi(g)$ est une expression (les autres fonctions élémentaires usuelles telles que tan, sinh ou argth s'expriment à l'aide de ces fonctions).

Un premier théorème encourageant a été donné par Moore dans [51], on peut également le trouver dans [53] : si l'expression à évaluer est lipschitzienne (cf. définition ci-dessous), l'écart entre l'intervalle calculé et l'image de la fonction correspondante est proportionnel au diamètre de l'intervalle d'entrée. Cette condition de linéarité est classique en analyse numérique : on mesure souvent la précision du résultat calculé par rapport à la perturbation sur les données via une relation du premier ordre, en supposant que les termes d'ordre supérieur sont négligeables, et on définit la notion de conditionnement à partir de cette relation linéaire. La différence majeure entre analyse numérique usuelle et arithmétique par intervalles est qu'avec cette dernière les termes d'ordre supérieur ne sont pas négligés (sinon les résultats ne contiennent plus de façon garantie les résultats cherchés) mais pris en compte dans les termes linéaires.

Le caractère lipschitzien des expressions arithmétiques découle de celui des opérations et fonctions utilisées pour les construire, au moins dans certains domaines, et il suffit de s'assurer que l'on ne sort pas de ces domaines.

Définition : expression lipschitzienne. Une expression arithmétique est lipschitzienne en un intervalle $\boldsymbol{x}_0$ si pour toute sous-expression de la forme $h(\boldsymbol{x}_0)^\alpha$ avec $0 < \alpha < 1$ on a $h(\boldsymbol{x}_0) > 0$ et pour toute sous-expression de la forme $\varphi(h(\boldsymbol{x}_0))$ avec φ une fonction élémentaire (cf. la définition des expressions arithmétiques), φ est lipschitzienne sur un intervalle dont l'intérieur contient $h(\boldsymbol{x}_0)$.

Théorème. Soit f une fonction en n variables définie par une expression arithmétique, lipschitzienne au sens intervalle en $\boldsymbol{x}_0 \in I\!I\!R^n$, on a

$$q(f(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x})) = \mathcal{O}(w(\boldsymbol{x}))$$

et

$$\forall \boldsymbol{x} \subset \boldsymbol{x}_0,\ w(f(\boldsymbol{x})) \leq \lambda_f(\boldsymbol{x}_0) w(\boldsymbol{x})$$

où $\lambda_f(\boldsymbol{x}_0)$ est la constante de Lipschitz de f en $\boldsymbol{x}_0$, elle ne dépend que de f et de $\boldsymbol{x}_0$.

Si donc on dispose d'une expression arithmétique $\boldsymbol{f}$ qui est une extension intervalle de la fonction à évaluer f et si de plus cette expression est lipschitzienne en $\boldsymbol{x}$ le vecteur d'intervalles sur lequel on évalue f, alors le résultat de cette évaluation sera de largeur proportionnelle à celle de $\boldsymbol{x}$. Si on désire un encadrement plus fin de l'image de $\boldsymbol{x}$ par f, une première solution — qui peut être considérée comme naïve tant que l'on dispose de techniques plus efficaces mais sera souvent le dernier recours, cf. §3.2.3 et 3.3 — consiste à découper le pavé $\boldsymbol{x}$ en petits sous-pavés $\boldsymbol{x}_i$ de largeur inférieure à $\varepsilon/\lambda_f(\boldsymbol{x}_0)$ dont l'union est égale à $\boldsymbol{x}$ et à évaluer $\boldsymbol{f}$ sur chaque sous-pavé $\boldsymbol{x}_i$. L'image de $\boldsymbol{x}$ par f, $f(\boldsymbol{x})$, est incluse dans l'union des $\boldsymbol{f}(\boldsymbol{x}_i)$ et, si les $\boldsymbol{x}_i$ ont une largeur assez petite, on a pour chaque $\boldsymbol{x}_i$ une surestimation inférieure à ε, d'où finalement $q(f(\boldsymbol{x}), \bigcup_i \boldsymbol{f}(\boldsymbol{x}_i)) \leq \varepsilon$. Cependant une telle stratégie pour une fonction f à n variables implique une découpe de $\boldsymbol{x}$ en au moins $[w(\boldsymbol{x}).\lambda_f(\boldsymbol{x}_0)/\varepsilon]^n$ sous-pavés, c'est-à-dire en un nombre exponentiel en le nombre de variables. Cela rejoint le théorème de Gaganov sur la difficulté d'évaluer précisément une fonction, cf. §2.2.

D'autres solutions reposent sur l'utilisation des formules de Taylor-Lagrange (l'utilisation des fonctions pente ne sera quasiment pas abordée dans cet article) et surtout sur la constatation suivante : toute formule dans laquelle on calcule le plus possible avec des scalaires et le moins possible avec des intervalles conduit à de bons encadrements des résultats cherchés. Les fonctions d'inclusion correspondantes sont présentées au paragraphe suivant.

2.2.3 Évaluation de fonctions : formes centrées et formules de Taylor-Lagrange

Soit f une fonction de $\mathbb{R}^n$ dans $\mathbb{R}$ continue et différentiable, supposons que l'on cherche à encadrer $f(\boldsymbol{x})$ pour un pavé $\boldsymbol{x} \subset \mathbb{R}^n$. La formule de Taylor-Lagrange au premier ordre indique que pour tout $x \in \boldsymbol{x}$ et tout $y \in \boldsymbol{x}$,

$$\exists \xi \in]x;y[\text{ (ou }]y;x[\text{ si } y < x) \ / \ f(y) = f(x) + f'(\xi).(y-x).$$

À de rares exceptions près, la valeur de ξ est inconnue ; cependant un encadrement de $f'(\xi)$ est donné par $f'(]x;y[)$ (ou $f'(]y;x[)$ si $y < x$, mais par abus de notation nous le noterons aussi $f(]x;y[)$), si de plus on dispose d'une fonction d'inclusion $\boldsymbol{f}'$ pour f', on a l'inclusion suivante :

$$f(y) \in f(x) + f'(]x;y[).(y-z) \subset f(x) + \boldsymbol{f}'([x;y]).(y-x).$$

Comme ceci est vrai pour tout $x \in \boldsymbol{x}$ et pour tout $y \in \boldsymbol{x}$, on a pour $x \in \boldsymbol{x}$ fixé

$$f(\boldsymbol{x}) \subset f(x) + \boldsymbol{f}'(\boldsymbol{x}).(\boldsymbol{x}-x)$$

et finalement la fonction $\boldsymbol{f}_{TL1}$ définie par $\boldsymbol{f}_{TL1}(\boldsymbol{x}) = f(x) + \boldsymbol{f}'(\boldsymbol{x}).(\boldsymbol{x}-x)$ est une fonction d'inclusion pour f sur $\boldsymbol{x}$. Le choix de $x \in \boldsymbol{x}$ est arbitraire ; ce point s'appelle *centre* mais n'est pas nécessairement le milieu de $\boldsymbol{x}$ et la forme correspondante $\boldsymbol{f}_{TL1}$ est une *forme centrée*. Ce n'est pas une expression arithmétique puisqu'elle fait intervenir un point de l'intervalle $\boldsymbol{x}$ et qu'un point n'est pas une quantité accessible par les opérations et fonctions élémentaires par intervalles.

Le théorème suivant permet de majorer le surencadrement et de se persuader qu'une forme centrée est intéressante dès lors que l'intervalle $\boldsymbol{x}$ est petit.

Théorème [53]. La fonction d'inclusion $\boldsymbol{f}_{TL1}$ vérifie

$$0 \leq w(\boldsymbol{f}_{TL1}(\boldsymbol{x})) - w(f(\boldsymbol{x})) \leq q(\boldsymbol{f}_{TL1}(\boldsymbol{x}), f(\boldsymbol{x})) \leq \frac{1}{2} w(\boldsymbol{f}'(\boldsymbol{x})).w(\boldsymbol{x}).$$

Corollaire. Si la fonction d'inclusion $\boldsymbol{f}'$ de f' est lipschitzienne au sens intervalle, alors la fonction d'inclusion donnée par la formule de Taylor-Lagrange au premier ordre vérifie

$$q(\boldsymbol{f}_{TL1}(\boldsymbol{x}), f(\boldsymbol{x})) = \mathcal{O}(w(\boldsymbol{x})^2).$$

On a donc une propriété d'*approximation quadratique* en utilisant comme fonction d'inclusion la formule de Taylor-Lagrange d'ordre 1. Une telle propriété signifie que si $\boldsymbol{x}$ est petit, alors cette formule donne de meilleurs résultats qu'une fonction d'inclusion naturelle pour laquelle l'approximation est seulement linéaire ; en revanche lorsque $\boldsymbol{x}$ est large, la fonction d'inclusion naturelle fournira un meilleur surencadrement que la forme centrée. La détermination de la largeur seuil w_0 à partir de laquelle l'une des deux formules est meilleure que l'autre est un problème non résolu. Une solution pratique consiste à évaluer $f(\boldsymbol{x})$ à l'aide d'une fonction d'inclusion naturelle et d'une forme centrée et à prendre l'intersection des deux encadrements obtenus, cependant cette solution est coûteuse en nombre d'opérations.

Revenons au choix du point x par rapport auquel on effectue le développement de Taylor : il est possible de choisir x de façon à maximiser la borne gauche du résultat, ou bien à minimiser la borne droite, ou bien à minimiser la largeur de l'évaluation (théorème de Baumann [8]). Le choix qui conduit à une largeur minimale consiste à prendre pour x le milieu de l'intervalle $\boldsymbol{x}$. Ce choix présente de plus l'avantage de fournir une fonction d'inclusion monotone pour l'inclusion, comme le précise le théorème suivant, dû à Caprani et Madsen [15].

Théorème. Si on prend comme fonction d'inclusion pour f sur $\boldsymbol{x}$ la fonction $\boldsymbol{f}_{TL1}$ donnée par le développement de Taylor-Lagrange à l'ordre 1 au point $x = \text{mid}(\boldsymbol{x})$ et si de plus la fonction d'inclusion $\boldsymbol{f}'$ pour f' est monotone pour l'inclusion, alors la fonction d'inclusion $\boldsymbol{f}_{TL1}$ est monotone pour l'inclusion.

Ce n'est plus vrai si le point utilisé pour le développement n'est pas le milieu de l'intervalle, comme l'illustre l'exemple suivant dans lequel on développe par rapport à une extrémité quelconque : soit $f(\boldsymbol{x}) = \boldsymbol{x}^2 - 2\boldsymbol{x} + 1$, sa dérivée est $f'(\boldsymbol{x}) = 2\boldsymbol{x} - 2$ et on prend pour $\boldsymbol{f}'$ la fonction d'inclusion naturelle $\boldsymbol{f}'(\boldsymbol{x}) = 2\boldsymbol{x} - 2$;

si pour $\boldsymbol{x} = [0;1]$ on développe en $x = 1$ et pour $\boldsymbol{y} = [0;0,9] \subset \boldsymbol{x}$ on développe en $y = 0$, on obtient

$$\begin{array}{rcl} f(\boldsymbol{x}) & \subset & f(1) + (2 \times [0;1] - 2) \times ([0;1] - 1) \\ & \subset & 0 + [-2;0] \times [-1;0] \\ & \subset & [0;1] \end{array}$$

et

$$\begin{array}{rcl} f(\boldsymbol{y}) & \subset & f(0) + (2 \times [0;0,9] - 2) \times ([0;0,9] - 0) \\ & \subset & 1 + [-2;-0,2] \times [0;0,9] \\ & \subset & [-0,8;1] : \end{array}$$

on n'a pas inclusion du second intervalle dans le premier.

On conserve également la monotonie de l'inclusion lorsque l'on évalue $f(\boldsymbol{x})$ en prenant l'intersection de l'évaluation à l'aide d'une fonction d'inclusion naturelle et du développement de Taylor-Lagrange à l'ordre 1 en $\mathrm{mid}(\boldsymbol{x})$, puisque chacune des deux fonctions d'inclusion l'est : en effet si $\boldsymbol{f}_1$ et $\boldsymbol{f}_2$ sont deux fonctions monotones pour l'inclusion et si $\boldsymbol{y} \subset \boldsymbol{x}$, alors $\boldsymbol{f}_1(\boldsymbol{y}) \cap \boldsymbol{f}_2(\boldsymbol{y}) \subset \boldsymbol{f}_1(\boldsymbol{y}) \subset \boldsymbol{f}_1(\boldsymbol{x})$, $\boldsymbol{f}_1(\boldsymbol{y}) \cap \boldsymbol{f}_2(\boldsymbol{y}) \subset \boldsymbol{f}_2(\boldsymbol{y}) \subset \boldsymbol{f}_2(\boldsymbol{x})$ et donc $\boldsymbol{f}_1(\boldsymbol{y}) \cap \boldsymbol{f}_2(\boldsymbol{y}) \subset \boldsymbol{f}_1(\boldsymbol{x}) \cap \boldsymbol{f}_2(\boldsymbol{x})$.

Une idée naturelle consiste à s'intéresser aux développements de Taylor avec restes de Lagrange d'ordres plus élevés, dans l'espoir d'obtenir des approximations d'ordre supérieur à 2. Une première remarque est qu'il n'est pas possible d'atteindre un ordre strictement supérieur à 2 si l'ensemble des opérations algébriques ne contient que $+$, $-$, $*$ et $/$: l'opération manquante, celle sans laquelle l'ordre 3 est inaccessible, est l'élévation au carré [29]. Par bonheur elle était incluse dans notre boîte à outils d'opérations et de fonctions, cf. §2.1, mais cela peut ne pas suffire... Une seconde constatation est que, pour des fonctions à n variables, il devient rapidement difficile d'exprimer les différentielles, ce qui explique qu'en pratique on n'utilise que des développements de Taylor d'ordres 1 ou 2. Enfin, pour ces mêmes fonctions, il est difficile d'exprimer le développement de Taylor à l'ordre 2 en n'utilisant que des élévations au carré, ce qui empêche d'obtenir une approximation cubique.

La formule de Taylor-Lagrange à l'ordre 2 est, pour toute fonction $f : \mathbb{R}^n \to \mathbb{R}$ de classe $\mathcal{C}^1$ et différentiable deux fois,

$$\forall x,\ \forall y,\ \exists \xi \in]x;y[\ /\ f(y) = f(x) + f'(x).(y-x) + \left(\frac{f''(\xi)}{2}.(y-x)\right).(y-x),$$

ce qui conduit à l'inclusion suivante pour un intervalle $\boldsymbol{x}$ donné, si $\boldsymbol{f}''$ est une fonction d'inclusion pour f'' :

$$\begin{array}{l} \forall x \in \boldsymbol{x},\ \forall y \in \boldsymbol{x}, \\ \begin{array}{rcl} f(y) & \in & f(x) + f'(x).(y-x) + \left(\frac{f''(\boldsymbol{x})}{2}.(y-x)\right).(y-x) \\ & \subset & \boldsymbol{f}_{TL2}(\boldsymbol{x}) := f(x) + f'(x).(\boldsymbol{x}-x) + \left(\frac{f''(\boldsymbol{x})}{2}.(\boldsymbol{x}-x)\right).(\boldsymbol{x}-x), \end{array} \end{array}$$

où ici encore x est appelé *centre* du développement. Si on se restreint aux fonctions à une seule variable, on peut donner une expression utilisant l'élévation au carré, qui est la suivante :

$$f(\boldsymbol{x}) \subset \boldsymbol{f}_{TL2}(\boldsymbol{x}) := f(x) + f'(x).(\boldsymbol{x}-x) + \frac{\boldsymbol{f}''(\boldsymbol{x})}{2}.(\boldsymbol{x}-x)^2.$$

La fonction d'inclusion $\boldsymbol{f}_{TL2}$ ainsi construite en prenant $x = \mathrm{mid}(\boldsymbol{x})$ est une approximation cubique de $f(\boldsymbol{x})$ si $\boldsymbol{f}''$ est lipschitzienne en $\boldsymbol{x}$, autrement dit

$$q(\boldsymbol{f}_{TL2}(\boldsymbol{x}), f(\boldsymbol{x})) = \mathcal{O}(w(\boldsymbol{x})^3).$$

Cette fonction d'inclusion peut être vue comme une interpolation de Hermite de degré 2 en $\check{x} = \mathrm{mid}(\boldsymbol{x})$. L'interpolation construite à partir des extrémités de $\boldsymbol{x}$ est également une approximation cubique si $\boldsymbol{f}''$ est lipschitzienne et elle présente de plus l'intérêt de fournir l'image exacte $f(\boldsymbol{x})$ dans le cas où f est de classe $\mathcal{C}^2$ et monotone sur $\boldsymbol{x}$.

Dans le cas général d'une fonction à plusieurs variables, il paraît difficile d'obtenir une approximation d'ordre strictement supérieur à 2.

Les deux fonctions d'inclusion obtenues à partir des développements de Taylor-Lagrange d'ordres 1 et 2 sont des cas particuliers des formes centrées, qui expriment l'écart entre $f(y)$ pour $y \in \boldsymbol{x}$ et $f(x)$ où x est un point fixé de $\boldsymbol{x}$, appelé centre. On peut par exemple utiliser la fonction de pente entre y et x pour obtenir une telle forme, en se basant sur l'égalité

$$f(y) = f(x) + f[y,x].(y-x)$$

avec $f[y,x]$ la fonction pente définie par

$$f[y,x] = \begin{cases} \frac{f(y)-f(x)}{y-x} & \text{si } y \neq x \\ f'(x) & \text{si } y = x \text{ (par continuité)} \end{cases}$$

on obtient la forme centrée

$$\boldsymbol{f}_p(\boldsymbol{x}) := f(x) + \boldsymbol{f}[\boldsymbol{x},x].(\boldsymbol{x}-x)$$

qui satisfait également la propriété d'approximation quadratique si $\boldsymbol{f}[\boldsymbol{x},x]$ est lipschitzienne en $\boldsymbol{x}$, mais pas la propriété de monotonie pour l'inclusion. La valeur $f[y,x]$ est une différence divisée ; dans le cas particulier où f est un polynôme donné sous forme développée, l'évaluation de f par le schéma de Horner fournit pour peu d'opérations supplémentaires cette différence divisée [70]. De plus cette fonction d'inclusion faisant intervenir plus de calculs avec des scalaires que $\boldsymbol{f}'(\boldsymbol{x})$ fournit en général des résultats plus précis que la formule de Taylor-Lagrange d'ordre 1.

2.3 Implantation sur ordinateur

Le plus souvent les intervalles sont représentés par leurs extrémités. La représentation par le centre et le rayon permet d'atteindre une plus grande précision, similaire à celle obtenue en utilisant des expansions de type « double-double » (cf. [59, 22] — pour une utilisation des double-double, cf. [45]), puisque le rayon peut être très petit par rapport au centre. En revanche, la définition des opérations arithmétiques avec cette représentation conduit à des intervalles 1,5 fois plus larges que ceux obtenus avec la représentation par leurs extrémités et cette surestimation est d'autant plus importante que le rayon est grand [67]. Dans cet article, Rump reporte des expérimentations en algèbre linéaire essentiellement, pour lesquelles les résultats obtenus avec les deux représentations sont comparables quand le rayon est faible (de l'ordre de 10^{-6} fois le centre) mais pas quand les intervalles d'entrée sont larges.

Jusqu'à présent, une arithmétique idéale a été utilisée pour écrire les formules. Une réalisation effective doit prendre en compte l'arithmétique disponible sur ordinateur, à savoir l'arithmétique en virgule flottante. En particulier, il faut éviter que les erreurs d'arrondi ne fassent perdre la propriété d'encadrement garanti des résultats. Si l'arithmétique utilisée est l'arithmétique flottante satisfaisant la norme IEEE-754, il est possible d'assurer cette propriété, au moins pour les opérations arithmétiques et algébriques. En effet cette norme impose que le résultat d'un calcul arithmétique soit l'arrondi du résultat exact, avec quatre modes d'arrondis disponibles : vers $+\infty$, vers $-\infty$, au plus près et vers 0. L'utilisation de l'arrondi vers $-\infty$ pour le calcul de la borne inférieure et de l'arrondi vers $+\infty$ pour celui de l'extrémité supérieure permet d'implanter l'arithmétique par intervalles ; on parle dans ce cas d'arrondi vers l'extérieur ou *outward rounding* en anglais. Cependant le changement de mode d'arrondi n'est pas toujours accessible à partir des langages de programmation de haut niveau et s'effectue souvent à l'aide de routines en assembleur ; il est de plus assez coûteux en temps car il impose de briser une exécution pipelinée des calculs.

La norme IEEE-754 définit aussi les valeurs $+\infty$ et $-\infty$, ainsi que leurs inverses $+0$ et -0, ce qui permet de définir l'inverse de $[+0;1]$ comme $[1;+\infty[$ par exemple et donc de manipuler des intervalles non bornés. Elle précise également que le résultat calculé d'une racine carrée est l'arrondi du résultat exact, mais ne spécifie rien pour les fonctions élémentaires (voir [44] pour une explication). Selon les implantations, les solutions adoptées vont d'une utilisation sans précaution des fonction élémentaires fournies par le langage à un remplacement des fonctions élémentaires par des procédures qui les évaluent en majorant l'erreur et ajoutent/soustraient cette erreur aux bornes du résultat (FI_LIB [31]), en passant par une correction par inflation (multiplication par $[1-\varepsilon;1+\varepsilon]$) et addition (de $[-\eta;+\eta]$) dans l'espoir de rattraper les erreurs (PROFIL/BIAS [39]), pour plus de détails, cf. §4.

Il est donc possible d'utiliser une bibliothèque d'arithmétique par intervalles qui fournit des encadrements garantis des résultats. Pour conserver cette propriété, il faut effectuer non seulement les calculs par intervalles, mais également les calculs scalaires, tels que le terme $f(x)$ apparaissant dans la formule de Taylor au premier ordre, en arithmétique par intervalles avec arrondi vers l'extérieur. En effet, rien n'assure que $f(x)$ est un nombre représentable en machine. Si f est lipschitzienne (au sens intervalle), alors la surestimation induite par le calcul à précision limitée de $f(\boldsymbol{x})$, où $\boldsymbol{x}$ peut aussi bien être un point qu'un intervalle, est bornée par $c_f u$ où c_f est une constante dépendant de f et u est la précision machine (cf. [53]).

Une autre approche de l'arithmétique par intervalles est une approche ensembliste, dans laquelle les intervalles ne constituent qu'une incarnation facile à manipuler des ensembles de réels. Dans une telle approche, la structure de treillis pour l'intersection et l'union est privilégiée et en particulier l'ensemble vide doit être manipulable, ce qui n'est pas prévu dans les définitions du §2.1. On trouve cependant dans [34] une proposition d'implantation de cette approche ensembliste, basée sur la norme IEEE-754 pour l'arithmétique flottante.

Une bibliothèque d'arithmétique par intervalles capable de mettre en œuvre les techniques exposées jusqu'à présent doit proposer un mécanisme de différentiation des fonctions à évaluer. Ce point ne sera pas détaillé ici. En bref, on peut penser à des techniques d'estimation numérique (par des schémas aux différences), mais elles donnent des encadrements beaucoup trop larges, ou encore à un pré-processeur produisant symboliquement, à partir de l'expression d'une fonction, une expression pour ses dérivées. Cette solution permet de dériver les fonctions en mode *backward*, c'est-à-dire de calculer les dérivées des sorties en fonction des variables intermédiaires, ce qui donne les résultats les plus précis. Il semble pourtant que la solution la plus couramment adoptée, parce que la plus simple à implanter, consiste à surcharger tous les opérateurs arithmétiques, les fonctions élémentaires etc. pour que soient calculées simultanément une valeur et les dérivées partielles correspondantes, en appliquant les règles usuelles de dérivation (règle de Leibniz, dérivée d'une composition de fonctions...) : on parle alors de différentiation automatique. Dans ce cas, on obtient des dérivées en mode *forward* ou dérivées en fonction des entrées uniquement.

2.4 Autres représentations pour le calcul ensembliste

Les intervalles ne forment pas une classe idéale d'ensembles pour le calcul garanti [40]. En particulier, pour les calculs dans $\mathbb{R}^n$, les pavés ne sont pas invariants par transformation affine du type $Ax + b$. Ces transformations sont pourtant couramment utilisées en arithmétique par intervalles puisque la formule de Taylor du premier ordre relève de cette catégorie.

Beaumont [9] remplace les pavés avec côtés parallèles aux axes par des parallélépipèdes rectangles « obliques » : il détermine les coins de ces pavés en caractérisant le problème à résoudre par des programmes linéaires, comme cela est fait pour $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b})$, fin du §3.1.1. Cette approche ne semble pas s'étendre naturellement aux problèmes dans lesquels interviennent des fonctions élémentaires par exemple.

Neumaier [54] propose d'utiliser des ellipsoïdes afin de fournir des résultats dont le volume ne déborde pas trop celui des résultats exacts ; il montre comment effectuer les opérations, en conservant un maximum de calculs scalaires pour limiter cette augmentation de volume, et présente quelques expérimentations illustrant le gain apporté par le calcul avec des ellipsoïdes.

Une étude menée par Kreinovich et Wolff von Gudenberg [40] sur le calcul ensembliste dans $\mathbb{C}$ aboutit à la conclusion que les ensembles dont les frontières sont définies — par morceaux — par des équations à au plus d paramètres et qui sont invariants par transformations affines dans $\mathbb{C}$ sont, dans le cas où d vaut 3 (ce qui est la plus petite valeur possible) les ensembles dont les frontières sont des segments de droite. Les rectangles sont des cas particuliers de tels ensembles, ils ne sont pas invariants par transformation affine mais autorisent des calculs simples.

3 Quelques algorithmes numériques revisités

Comme cela a été expliqué aux §2.1 et 2.2, l'arithmétique par intervalles ne présente pas les propriétés arithmétiques usuelles et les calculs ont une fâcheuse propension à surestimer les résultats. En particulier l'utilisation directe des algorithmes usuels ne donne pas toujours des résultats satisfaisants. C'est ce que nous nous attacherons à mettre en évidence dans cette partie.

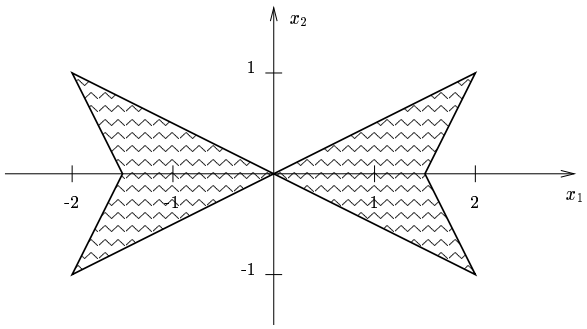

FIG. 3 – *Solution d'un système linéaire intervalle.*

3.1 Résolution de systèmes linéaires

Cette partie doit beaucoup à la thèse de Beaumont [9] pour les aspects de complexité (même si ses travaux sur l'encadrement des solutions par des boîtes obliques et donc de volume réduit sont totalement occultés) et à l'ouvrage de Neumaier [53] pour les algorithmes et leur analyse. Ils ne seront donc pas cités systématiquement par la suite.

3.1.1 Définition du problème et résultats de complexité

Résoudre un système linéaire signifie, en arithmétique usuelle, se donner une matrice A de taille $n \times n$ et un vecteur $b \in \mathbb{R}^n$ et déterminer un vecteur $x \in \mathbb{R}^n$ tel que $Ax = b$. Si on se donne une matrice intervalle de taille $n \times n$ $\boldsymbol{A}$ et un vecteur intervalle $\boldsymbol{b} \in I\mathbb{R}^n$, il n'existe pas nécessairement de vecteur $\boldsymbol{x} \in I\mathbb{R}^n$ vérifiant [1] $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.

Une définition classique du problème de la résolution de systèmes linéaires par intervalles consiste à déterminer l'ensemble des vecteurs $x \in \mathbb{R}^n$ qui sont solution d'un système linéaire ponctuel $Ax = b$ avec $A \in \boldsymbol{A}$ et $b \in \boldsymbol{b}$, ou plus précisément un pavé contenant cet ensemble qui n'est pas nécessairement convexe, comme le montre l'exemple suivant [53] : l'ensemble des $x \in \mathbb{R}^2$ pour lesquels

$$\exists A \in \boldsymbol{A} = \begin{pmatrix} [2;4] & [-1;1] \\ [-1;1] & [2;4] \end{pmatrix} \text{ et } \exists b \in \boldsymbol{b} = \begin{pmatrix} [-3;3] \\ 0 \end{pmatrix} \text{ tels que } Ax = b$$

est représenté sur la figure 3.

Dans la suite, en notant Conv E l'enveloppe convexe d'un ensemble E, nous chercherons à déterminer un pavé contenant

$$\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b}) = \text{Conv}\{x \in \mathbb{R}^n \,/\, \exists A \in \boldsymbol{A}, \exists b \in \boldsymbol{b}, Ax = b\}.$$

Grâce au théorème d'Oettli et Pragger [57], l'ensemble $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b})$ peut être caractérisé de façon équivalente par

$$x \in \Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b}) \Leftrightarrow |\text{mid}(\boldsymbol{A})x - \text{mid}(\boldsymbol{b})| \leq \text{rad}(\boldsymbol{A})|x| + \text{rad}(\boldsymbol{b}).$$

On peut réécrire ce théorème de la façon suivante : pour toute matrice diagonale D telle que $|D| = I_n$,

$$\begin{array}{l} x \in \Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b}) \cap \{y \in \mathbb{R}^n \,/\, Dy = |y|\} \\ \Leftrightarrow \left\{ \begin{array}{ccc} D & x & \geq & 0 \\ \begin{pmatrix} \text{mid}(\boldsymbol{A}) - \text{rad}(\boldsymbol{A})D \\ -\text{mid}(\boldsymbol{A}) - \text{rad}(\boldsymbol{A})D \end{pmatrix} & x & \geq & \begin{pmatrix} \bar{b} \\ -\underline{b} \end{pmatrix} \end{array} \right. \end{array}$$

et un algorithme de calcul de $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b})$ consiste à résoudre, pour chaque quadrant défini à l'aide d'une matrice D par $\{y \in \mathbb{R}^n / Dy = |y|\}$, les programmes linéaires définis pour tout $i \in \{1 \ldots n\}$ par les contraintes

1. Si par exemple $\boldsymbol{A} = [-1;1]$ et $\boldsymbol{b} = [2;3]$, le produit de $\boldsymbol{A}$ par tout intervalle $\boldsymbol{x}$ contiendra 0 puisque $0 \in \boldsymbol{A}$ et ne pourra donc pas être égal à $\boldsymbol{b}$.

ci-dessus et la fonction économique min $\underline{x_i}$ ou max $\overline{x_i}$. Chaque programme linéaire peut être résolu en temps polynomial [36] et il y a $2n.2^n$ programmes linéaires de ce type : cet algorithme, dû à Aberth [1] (et qui remonte même à Oettli [56]) a une complexité exponentielle en temps. On ne peut guère faire mieux, puisque Rohn et Kreinovich [63] ont montré que la détermination de $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b})$ est un problème NP-dur.

D'autres définitions consistent à utiliser des quantificateurs universels et existentiels, comme par exemple

$$\Sigma_{\forall\exists}(\boldsymbol{A}, \boldsymbol{b}) = \mathrm{Conv}\{x \in I\!R^n \,/\, \forall A \in \boldsymbol{A}, \exists b \in \boldsymbol{b}, Ax = b\}$$

ou

$$\Sigma_{\exists\forall}(\boldsymbol{A}, \boldsymbol{b}) = \mathrm{Conv}\{x \in I\!R^n \,/\, \forall b \in \boldsymbol{b}, \exists A \in \boldsymbol{A}, Ax = b\}.$$

Lakeyev [42] a ensuite montré que si le problème est défini en utilisant un quantificateur par élément de $\boldsymbol{A}$ et de $\boldsymbol{b}$ et si le nombre de quantificateurs existentiels croît plus rapidement qu'une puissance en la dimension n, alors le problème est NP-complet. Un dernier résultat de complexité est que si l'on cherche le plus petit pavé englobant $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b})$ à $1/4n^4$ près, alors ce problème est encore NP-dur [62]. Autrement dit, tout comme pour le problème de l'évaluation de l'image d'une fonction sur un intervalle, l'objectif est de déterminer un surencadrement « pas trop large » de $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b})$.

3.1.2 Élimination de Gauss par intervalles

L'algorithme le plus connu pour la résolution de systèmes linéaires est probablement l'algorithme d'élimination de Gauss. On peut tenter de l'appliquer à un système linéaire intervalle $\boldsymbol{Ax} = \boldsymbol{b}$, aussi longtemps qu'aucun pivot ne contient 0. Dans l'hypothèse où l'élimination a été poursuivie jusqu'à son terme, on obtient à la place de $\boldsymbol{A}$ une matrice triangulaire supérieure $\boldsymbol{U}$ et on peut construire une matrice triangulaire inférieure $\boldsymbol{L}$ contenant les coefficients des combinaisons linéaires de lignes effectuées pendant l'élimination. Il est à noter que l'on a simplement l'inclusion $\boldsymbol{A} \subset \boldsymbol{LU}$ puisque $\boldsymbol{LU} = \{L \times U \,/\, L \in \boldsymbol{L}, U \in \boldsymbol{U}\}$ contient des produits $L \times U$ où L et U ne proviennent pas de la factorisation d'une même matrice $A \in \boldsymbol{A}$. On peut alors trouver un surencadrement de $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b})$ en « résolvant » par descente triangulaire le système linéaire $\boldsymbol{Ly} = \boldsymbol{b}$ puis par remontée $\boldsymbol{Ux} = \boldsymbol{y}$. En effet, pour tout $z \in \Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b})$, $\exists A \in \boldsymbol{A}$ et $\exists b \in \boldsymbol{b}$ vérifiant $Az = b$ et A admet une factorisation $A = L \times U$ avec $L \in \boldsymbol{L}$ et $U \in \boldsymbol{U}$, autrement dit $LUz = b \in \boldsymbol{b}$, d'où $Uz \in \boldsymbol{U}z \subset \boldsymbol{y}$ solution de $\boldsymbol{Ly} = \boldsymbol{b}$ et finalement $z \in \boldsymbol{x}$ solution de $\boldsymbol{Ux} = \boldsymbol{y}$, donc $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b}) \subset \boldsymbol{x}$ obtenu par élimination de Gauss et remontée triangulaire.

On ne connaît pas de condition nécessaire et suffisante sur $\boldsymbol{A}$ garantissant que l'on peut effectuer une élimination de Gauss sans avoir de pivot contenant 0. Appliquer une stratégie de pivot partiel ne permet pas nécessairement de stabiliser l'algorithme de Gauss, comme l'illustre l'exemple suivant :

$$\boldsymbol{A} = \begin{pmatrix} [0,1;0,9] & 0 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{pmatrix}$$

admet la factorisation LU suivante, obtenue sans pivotage :

$$\boldsymbol{L} = \begin{pmatrix} 1 & 0 & 0 \\ [-10;-1/0,9] & 1 & 0 \\ 0 & -1/3 & 1 \end{pmatrix}, \quad \boldsymbol{U} = \begin{pmatrix} [0,1;0,9] & 0 & 0 \\ 0 & 3 & -1 \\ 0 & 0 & 5/3 \end{pmatrix},$$

alors que la stratégie de pivot partiel conduit à échanger les deux premières lignes et après la première étape d'élimination on obtient :

$$\begin{pmatrix} -1 & 3 & -1 \\ 0 & [0,3;2,7] & [-0,9;-0,1] \\ 0 & -1 & 2 \end{pmatrix} \ni \begin{pmatrix} -1 & 3 & -1 \\ 0 & 0,4 & -0,8 \\ 0 & -1 & 2 \end{pmatrix}$$

où le bloc 2×2 en bas à droite est non inversible.

Comme en algorithmique numérique classique, il existe peu de résultats positifs caractérisant les matrices pour lesquelles l'élimination de Gauss peut être conduite à son terme, ils concernent essentiellement des sous-classes de matrices telles que les M-matrices qui interviennent souvent lors de la discrétisation d'EDP par exemple, ou les H-matrices, pour lesquelles on s'affranchit des contraintes de signe définissant les M-matrices.

Une M-matrice $\boldsymbol{A} \in I\!I\!R^{n\times n}$ est une matrice dont les éléments hors-diagonaux $\boldsymbol{A}_{i,j}$ sont négatifs pour $i \neq j$ et telle qu'il existe un vecteur strictement positif $u \in I\!R^n$ vérifiant $\boldsymbol{A}u > 0$. Une M-matrice est régulière (c'est-à-dire qu'elle ne contient que des matrices inversibles). Une caractérisation équivalente est qu'une M-matrice a tous ses éléments hors-diagonaux ≤ 0 et son inverse a tous ses éléments positifs. Enfin, $\boldsymbol{A}$ est une M-matrice ssi $\underline{\boldsymbol{A}}$ et $\overline{\boldsymbol{A}}$ sont des M-matrices[2] et même $\boldsymbol{A}^{-1} = [\overline{\boldsymbol{A}}^{-1}; \underline{\boldsymbol{A}}^{-1}]$.

Une H-matrice $\boldsymbol{A} \in I\!I\!R^{n\times n}$ est une matrice dont la matrice de comparaison $< \boldsymbol{A} >$, définie par $< \boldsymbol{A} >_{i,i} = \mathrm{mig}(\boldsymbol{A}_{i,i})$ et $< \boldsymbol{A} >_{i,j} = -|\boldsymbol{A}_{i,j}|$ pour $i \neq j$, est une M-matrice ou de façon équivalente pour laquelle il existe un vecteur $u > 0$ tel que $< \boldsymbol{A} > u$ a toutes ses composantes strictement positives. Les matrices à diagonale strictement dominante sont des H-matrices pour lesquelles le vecteur u est égal à $(1, \ldots 1)^t$.

L'algorithme d'élimination de Gauss peut être appliqué aux H-matrices sans problème de division par un pivot contenant 0. De plus, si $\boldsymbol{A}$ est une M-matrice, alors pour tout second membre $\boldsymbol{b} \in I\!I\!R^n$, le vecteur intervalle $\boldsymbol{x}$ obtenu par l'algorithme d'élimination de Gauss vérifie $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b}) \subset \boldsymbol{x} \subset \boldsymbol{A}^{-1}\boldsymbol{b}$.

L'algorithme d'élimination de Gauss par intervalles peut conduire à des surencadrements du résultat beaucoup trop larges,. Illustrons ce phénomène par un exemple :

$$\text{soient } \boldsymbol{A} = \begin{pmatrix} 1 & & 0 \\ \vdots & \ddots & \\ 1 & \ldots & 1 \end{pmatrix} \text{ et } \boldsymbol{b} = \begin{pmatrix} [-\alpha;\alpha] \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

l'algorithme de descente triangulaire conduit à $\boldsymbol{x} = ([-\alpha;\alpha], [-\alpha;\alpha], [-2\alpha;2\alpha], \ldots\ [-2^{n-2}\alpha; 2^{n-2}\alpha])^t$ alors que $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b}) = ([-\alpha;\alpha], [-\alpha;\alpha], 0, \ldots 0)^t$. On a dans ce cas une surestimation exponentielle du résultat.

Cet algorithme est revenu en usage après qu'un préconditionnement lui a été adjoint. L'idée d'un préconditionnement consiste à rapprocher la matrice du système linéaire à résoudre de la matrice identité. On peut songer *a priori* à préconditionner un système linéaire $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ à droite et à gauche par des matrices scalaires C et C' et à résoudre $C\boldsymbol{A}C'\boldsymbol{z} = C\boldsymbol{b}$, la solution étant $\boldsymbol{x} = C'\boldsymbol{z}$; dans le cas où $\mathrm{mid}(\boldsymbol{A})\boldsymbol{A}$ est régulière, on dispose d'une borne pour $|\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b})|$ qui est rendue minimale par le choix $C = \mathrm{mid}(\boldsymbol{A})^{-1}$, $C' = I$ qui est le préconditionnement le plus utilisé en pratique.

Dans le cas où $\mathrm{mid}(\boldsymbol{A})^{-1} \times \boldsymbol{A}$ est régulière, l'algorithme d'élimination de Gauss peut lui être appliqué. Si de plus $\mathrm{mid}(\boldsymbol{A})^{-1} \times \boldsymbol{A}$ est à diagonale strictement dominante, alors la solution obtenue par élimination de Gauss et remontée triangulaire est une approximation quadratique de $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b})$. Cependant, Mayer et Rohn [48] ont montré que dans le cas où la matrice milieu de cette matrice est diagonale, alors l'algorithme de Hansen-Bliek-Rohn-Ning-Kearfott [55] donne exactement $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b})$.

Pour résumer, l'algorithme d'élimination de Gauss était connu pour donner des surestimations très larges des résultats, avant que l'on ne s'aperçoive expérimentalement qu'il se comportait bien sur des systèmes préconditionnés puis qu'il soit prouvé optimal pour les M-matrices.

3.1.3 Méthode itérative de Gauss-Seidel par intervalles

Une autre façon d'aborder le problème, qui peut être comparée à une technique de propagation de contraintes, consiste à « résoudre » la i-ème ligne du système linéaire $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ en la i-ème variable $\boldsymbol{x}_i$: si un pavé de recherche est donné, la i-ème contrainte peut se réécrire en

$$\boldsymbol{x}_i' = \left(\boldsymbol{b}_i - \sum_{j\neq i} \boldsymbol{A}_{i,j}\boldsymbol{x}_j\right) / \boldsymbol{A}_{i,i} \cap \boldsymbol{x}_i.$$

On en déduit l'itération série de Gauss-Seidel où k est le numéro de l'itération :

$$\boldsymbol{x}_i^{(k+1)} = \left(\boldsymbol{b}_i - \sum_{j=1}^{i-1} \boldsymbol{A}_{i,j}\boldsymbol{x}_j^{(k+1)} - \sum_{j=i+1}^{n} \boldsymbol{A}_{i,j}\boldsymbol{x}_j^{(k)}\right) / \boldsymbol{A}_{i,i} \cap \boldsymbol{x}_i^{(k)},$$

2. Cette dernière caractérisation donne lieu à un critère effectif car en général les problèmes traités en arithmétique par intervalles sont de petite taille, comparée à celle des problèmes résolus en utilisant l'arithmétique flottante usuelle, et l'inversion de ces deux matrices ponctuelles à l'aide d'une procédure numérique en précision finie n'est pas déraisonnable.

que l'on peut aussi écrire de façon matricielle par

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{M}^{-1}\left(\boldsymbol{b} + \boldsymbol{N}\boldsymbol{x}^{(k)}\right) \cap \boldsymbol{x}^{(k)}$$

avec $\boldsymbol{M}$ la partie triangulaire inférieure de $\boldsymbol{A}$, diagonale incluse, et $\boldsymbol{N}$ l'opposé de la partie triangulaire supérieure de $\boldsymbol{A}$, diagonale exclue (on a $\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$).

Si cette itération converge, sa limite est le point fixe de l'équation

$$\boldsymbol{x} = \boldsymbol{M}^{-1}(\boldsymbol{b} + \boldsymbol{N}\boldsymbol{x})\,;$$

de plus, si le pavé initial $\boldsymbol{x}^{(0)} \supset \Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b})$, alors tous les itérés contiennent $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b})$ et la suite $(\boldsymbol{x}^{(k)})_k$ ainsi définie est monotone décroissante pour l'inclusion. La convergence est assurée si cette itération est contractante (cf. le théorème du point fixe [68], chap. XII), ce qui s'écrit dans ce cas $\rho(|\boldsymbol{M}|^{-1} \times |\boldsymbol{N}|) < 1$, avec ρ désignant le rayon spectral, et on a alors existence et unicité de la solution. En général cette condition est difficile à vérifier, mais, avec l'arithmétique par intervalles, le caractère contractant d'une itération peut facilement être vérifié en pratique, en testant l'inclusion d'un itéré dans le précédent.

Des variantes de ce théorème établissent la non convergence de l'algorithme itératif de Gauss-Seidel dans le cas où $\boldsymbol{A}$ n'est pas une H-matrice, qui se traduit par la stagnation de certaines bornes, puisque tout comportement divergent est exclus de par la monotonie de la suite $(\boldsymbol{x}^{(k)})_k$. Réciproquement, il y a convergence vers un surencadrement de $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b})$ dans le cas où $\boldsymbol{A}$ est une H-matrice ; si de plus $\boldsymbol{A}$ est une M-matrice, alors le point fixe est exactement $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b})$.

Ici aussi, un préconditionnement peut permettre de se ramener à un système linéaire dont la matrice est une H-matrice et le préconditionnement le plus utilisé consiste à multiplier le système à gauche par $\mathrm{mid}(\boldsymbol{A})^{-1}$. L'algorithme itératif de Gauss-Seidel ainsi préconditionné porte le nom d'algorithme de Rump [65]. Pour une étude des préconditionneurs pour l'algorithme de Gauss-Seidel, voir [6].

Les tentatives visant à adapter les techniques de *splitting* (découpage de $\boldsymbol{A}$ en $\boldsymbol{M} - \boldsymbol{N}$) plus générales, comme la méthode de Krawczyk, ou de surrelaxation, comme les méthodes SOR ou SSOR, se sont soldées par un échec ; en particulier la méthode de Gauss-Seidel est prouvée être la meilleure des méthodes de splitting.

Dans le cas où la matrice du système éventuellement préconditionné $\boldsymbol{A}$ est une H-matrice et où $\mathrm{mid}(\boldsymbol{A})$ est diagonale, alors l'algorithme d'élimination de Gauss fournit un meilleur surencadrement de $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b})$. Une utilisation pratique de Gauss-Seidel peut se limiter à quelques itérations pour améliorer le résultat de l'algorithme d'élimination de Gauss ou de Hansen-Bliek-Rohn-Ning-Kearfott.

Les méthodes flottantes de type Krylov [13] ne semblent pas avoir donné lieu à des méthodes par intervalles ; l'explication en est que ces méthodes construisent des bases (bi-)orthogonales des sous-espaces de Krylov et que la notion d'orthogonalité a peu de sens en arithmétique par intervalles.

3.2 Résolution de systèmes non linéaires

Les algorithmes de résolution de systèmes linéaires constituent la brique de base des méthodes de Newton pour la résolution de systèmes non linéaires, comme nous allons le montrer dans cette partie.

3.2.1 Présentation de la méthode de Newton par intervalles

La première méthode enseignée pour la résolution de systèmes non linéaires est la méthode de Newton. Elle présente des propriétés de convergence (convergence quadratique, méthode auto-correctrice) qui en font un bon point de départ pour construire une méthode par intervalles. Dans le cas d'une fonction $f : \mathbb{R} \to \mathbb{R}$, cette méthode consiste à prendre, pour approximation de la solution x^* telle que $f(x^*) = 0$, l'intersection de la tangente à la courbe de f avec l'axe des x. L'itération est donnée par

$$\begin{array}{l} x_0 \text{ fixé} \\ x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \text{ en supposant } f'(x_k) \text{ non nulle.} \end{array}$$

Si on remplace les valeurs ponctuelles par des quantités intervalles, on peut en déduire une itération qui est bien définie si $\boldsymbol{f}(\boldsymbol{x}_k) \not\ni 0 \ldots$ mais qui diverge, en effet la largeur des intervalles augmente : $w(\boldsymbol{x}_{k+1}) = w(\boldsymbol{x}_k) + w(\boldsymbol{f}(\boldsymbol{x}_k)/\boldsymbol{f}'(\boldsymbol{x}_k)) \geq w(\boldsymbol{x}_k)$. Il faut donc mettre au point une itération spécifique à l'arithmétique par

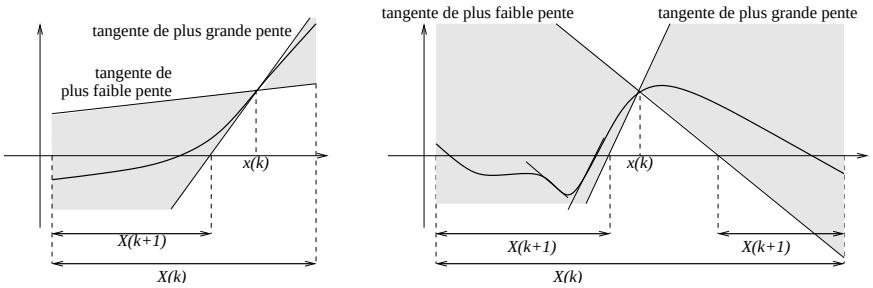

FIG. 4 – *Schéma de la méthode de Newton dans le cas à une seule variable.*

intervalles, dont le succès va reposer sur l'utilisation aussi fréquente que possible de quantités ponctuelles dans les formules. Le principe de la méthode de Newton par intervalles consiste à encadrer le graphe de f par un cône défini par les tangentes extrémales de f (cf. partie grisée, figure 4) et les zéros de f par l'intersection de ce cône avec l'axe des x.

Notons $\boldsymbol{x}$ l'intervalle dans lequel on recherche les zéros de f et $\tilde{x}$ un point quelconque de $\boldsymbol{x}$ qui permet de définir le cône et que l'on pourrait appeler *centre* par analogie avec le §2.2.3 ; on prend pour valeurs des pentes du cône les extrémités de $\boldsymbol{f}'(\boldsymbol{x})$ où $\boldsymbol{f}'$ est une extension intervalle de f'. Pour tout point (x, y) appartenant au cône, il existe $\alpha \in \boldsymbol{f}'(\boldsymbol{x})$ tel que

$$y = f(\tilde{x}) + \alpha(x - \tilde{x}).$$

En particulier pour tout point du cône sur l'axe des x, on a $y = 0$ et donc

$$\exists \alpha \in \boldsymbol{f}'(\boldsymbol{x}) \ / \ x = \tilde{x} - \frac{f(\tilde{x})}{\alpha}.$$

Finalement, l'intersection du cône avec l'axe des x est incluse dans $\tilde{x} - f(\tilde{x})/\boldsymbol{f}'(\boldsymbol{x})$. Comme on cherche les zéros de f appartenant à $\boldsymbol{x}$, on prend l'intersection de cet ensemble avec $\boldsymbol{x}$ et le nouvel ensemble de recherche est

$$\boldsymbol{x}' = \left(\tilde{x} - \frac{f(\tilde{x})}{\boldsymbol{f}'(\boldsymbol{x})}\right) \cap \boldsymbol{x}.$$

Si $\boldsymbol{f}'(\boldsymbol{x})$ contient 0, on effectue une division par intervalles étendue et après intersection avec $\boldsymbol{x}$ on obtient deux sous-intervalles, comme illustré par le schéma de droite de la figure 4.

Dans le cas où f est une fonction à plusieurs variables $f : \mathbb{R}^n \to \mathbb{R}^n$, la dérivée est remplacée par la Jacobienne de f et l'itération de Newton devient

$$\begin{array}{l} \boldsymbol{x}^0 \text{ donné} \\ \boldsymbol{x}^{k+1} = \left(\tilde{x} - \boldsymbol{f}'(\boldsymbol{x}_k)^{-1}.f(\tilde{x})\right) \cap \boldsymbol{x}^k. \end{array}$$

L'itération ainsi construite est clairement monotone décroissante pour l'inclusion puisque par construction $\boldsymbol{x}^{k+1} \subset \boldsymbol{x}^k$.

Notons $N(\boldsymbol{x}, \tilde{x})$ le pavé $\tilde{x} - \Sigma_{\exists\exists}(\boldsymbol{f}'(\boldsymbol{x}), f(\tilde{x}))$. L'existence, l'existence et unicité ou l'absence de zéro de f dans $\boldsymbol{x}$ sont données par le théorème suivant.

Théorème (Moore et Nickel).
Soit $f : \mathbb{R}^n \to \mathbb{R}^n$ et soit $\boldsymbol{f}$ une extension intervalle de f (définie au §2.2.2), si $\boldsymbol{f}$ est lipschitzienne (au sens intervalle) et si $\boldsymbol{f}'(\boldsymbol{x})$ ne contient pas 0, alors

- tout zéro de f dans $\boldsymbol{x}$ appartient aussi à $N(\boldsymbol{x}, \tilde{x})$ pour $\tilde{x} \in \boldsymbol{x}$;
- si $N(\boldsymbol{x}, \tilde{x}) \cap \boldsymbol{x} = \emptyset$ alors f n'admet pas de zéro dans $\boldsymbol{x}$;
- soit $\tilde{x} \in \text{int}(\boldsymbol{x})$ l'intérieur de $\boldsymbol{x}$, si $N(\boldsymbol{x}, \tilde{x}) \subset \boldsymbol{x}$ alors f admet au moins un zéro dans $\boldsymbol{x}$;
- soit $\tilde{x} \in \text{int}(\boldsymbol{x})$, si $N(\boldsymbol{x}, \tilde{x}) \subset \text{int}(\boldsymbol{x})$ alors f admet un seul zéro dans $\boldsymbol{x}$.

Les deux premières propriétés sont des propriétés de garantie des résultats et la troisième est une formulation par intervalle du théorème de Brouwer. Il est remarquable que les hypothèses de ce théorème soient faciles à vérifier en arithmétique par intervalles : comparer $\boldsymbol{x}$ et un surencadrement de $N(\boldsymbol{x}, \tilde{x})$ est typiquement ce que cette arithmétique rend possible. La quatrième propriété découle du théorème du point fixe contractant.

Avec une implantation sur ordinateur, il faut bien sûr calculer $f(\tilde{x})$ par intervalles et il peut être impossible de vérifier si $N(\boldsymbol{x}, \tilde{x})$ est inclus dans l'intérieur de $\boldsymbol{x}$, ne serait-ce qu'à cause de la précision machine limitée et des arrondis vers l'extérieur qui renforcent le surencadrement. Une technique fréquemment mise en œuvre depuis son introduction par Rump [64] pour la résolution de systèmes linéaires consiste à « gonfler » l'intervalle $\boldsymbol{x}$ en $\boldsymbol{x}_\varepsilon$, avec ε un paramètre caractérisant l'augmentation de largeur, et à tester l'inclusion de $N(\boldsymbol{x}_\varepsilon, \tilde{x})$ dans $\boldsymbol{x}_\varepsilon$. Différents choix pour $\boldsymbol{x}_\varepsilon$ sont par exemple $\boldsymbol{x}_\varepsilon = \boldsymbol{x} + w(\boldsymbol{x})[-\varepsilon;\varepsilon] + [-\eta;\eta]$ avec $\varepsilon = u$ la précision machine et η le plus petit nombre flottant strictement positif, ou bien $\boldsymbol{x}_\varepsilon = \boldsymbol{x} + w(\boldsymbol{x})[-\varepsilon;\varepsilon]$ avec $\varepsilon = 0,25$ pour que le « gonflage » soit plus net, cf. [47] pour différentes possibilités, des exemples d'applications et des théorèmes de point fixe dans le cas où la fonction f est une contraction de type particulier (P-contraction).

3.2.2 Cas où la fonction admet au plus un zéro

La mise en œuvre de l'itération de Newton nécessite la résolution d'un système linéaire dont la matrice est $\boldsymbol{f}'(\boldsymbol{x})$. On supposera pour commencer que f a au plus un zéro dans le pavé $\boldsymbol{x}$ et même que $\boldsymbol{f}'(\boldsymbol{x})$ est régulière. On sait que si $\tilde{x} = \mathrm{mid}(\boldsymbol{x})$ à chaque étape et si la résolution des systèmes linéaires de la forme $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ s'effectue en multipliant par $\boldsymbol{A}^I \supset \mathrm{Conv}\{A^{-1}, A \in \boldsymbol{A}\}$ où $\boldsymbol{A}^I$ est régulière, alors pour tout indice k pour lequel $\boldsymbol{x}^{k+1} \neq \emptyset$, on a $\mathrm{vol}(\boldsymbol{x}^{k+1}) \leq 1/2\ \mathrm{vol}(\boldsymbol{x}^k)$ avec $\mathrm{vol}(\boldsymbol{x})$ le volume du parallélépipède $\boldsymbol{x}$; le choix $\tilde{x} = \mathrm{mid}(\boldsymbol{x})$ sera adopté pour la suite. Une méthode couramment utilisée est la méthode de Gauss-Seidel préconditionnée par un préconditionneur noté C. L'itération de Newton ainsi définie est connue sous le nom d'itération de Hansen-Sengupta. On peut montrer que si $C\boldsymbol{A}$ est une H-matrice et si $\tilde{x} = \mathrm{mid}(\boldsymbol{x})$ alors l'itération de Hansen-Sengupta définit une suite qui soit converge vers l'unique zéro de f dans $\boldsymbol{x} = \boldsymbol{x}^0$, soit devient la suite stationnaire $\boldsymbol{x}^k = \emptyset$ à partir d'un certain rang. Une telle suite est dite *fortement convergente*. Le théorème de Moore et Nickel s'applique encore en remplaçant $\Sigma_{\exists\exists}(\boldsymbol{f}'(\boldsymbol{x}), f(\tilde{x}))$ par le résultat de l'algorithme de Gauss-Seidel. Dans ce cas également, le préconditionneur le plus fréquemment utilisé est l'inverse (numériquement approché) de la matrice milieu : en effet on connaît des caractérisations des H-matrices à partir de matrices $\boldsymbol{A}$ préconditionnées par $\mathrm{mid}(\boldsymbol{A})^{-1}$, cf. §3.1.

On peut utiliser toute autre méthode pour résoudre le système linéaire $\boldsymbol{f}'(\boldsymbol{x}).(\boldsymbol{x}' - \tilde{x}) = f(\tilde{x})$, comme la multiplication par une matrice « inverse » $\boldsymbol{A}^I$ vérifiant $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b}) \subset \boldsymbol{A}^I.\boldsymbol{b}$, éventuellement après préconditionnement du système : on parle alors d'itération de Newton généralisée pour l'itération définie par

$$\begin{array}{l} \boldsymbol{x}^0 \text{ donné} \\ \boldsymbol{x}^{k+1} = \left(\tilde{x}^k - \boldsymbol{f}'(\boldsymbol{x}^k)^I.f(\tilde{x}^k)\right) \cap \boldsymbol{x}^k. \end{array}$$

En général, la méthode de Hansen-Sengupta converge dans beaucoup de cas mais elle converge lentement, alors qu'une méthode de Newton généralisée converge rapidement si on est proche du zéro x^* de f dans $\boldsymbol{x}^0$. Pour tirer parti de ces deux types de comportements, on peut songer à une itération hybridant la méthode de Newton généralisée et celle de Hansen-Sengupta :

$$\begin{array}{lcl} \boldsymbol{x}^0 \text{ donné} & & \\ \boldsymbol{x}^{k+1/2} & = & \left(\tilde{x}^k - \boldsymbol{f}'(\boldsymbol{x}^k)^I.f(\tilde{x}^k)\right) \cap \boldsymbol{x}^k \\ \boldsymbol{x}^{k+1} & = & \left(\tilde{x}^k - \boldsymbol{f}'(\boldsymbol{x}^k)^{GS}.f(\tilde{x}^k)\right) \cap \boldsymbol{x}^{k+1/2}. \end{array}$$

On peut montrer la forte convergence de cette suite dans le cas où $\tilde{x}^k = \mathrm{mid}(\boldsymbol{x}^k)$ et où $\boldsymbol{f}'(\boldsymbol{x}^k)$ (ou bien $C\boldsymbol{f}'(\boldsymbol{x}^k)$ avec C préconditionneur) est une H-matrice. On peut même montrer que la convergence est au moins linéaire et sous des hypothèses plus fortes qu'elle est quadratique. Un autre choix possible pour $\tilde{x}^k$ consiste à utiliser un zéro approché de f déterminé par une procédure flottante, comme quelques pas de la méthode de Newton flottante par exemple. Enfin, d'autres techniques, inspirées de notions de consistance de la programmation logique sous contraintes, combinent itération de Newton à une seule variable dans chaque dimension et itération de Hansen-Sengupta [73].

3.2.3 Cas où la fonction admet plusieurs zéros

Quand la fonction f admet plusieurs zéros dans le pavé $\boldsymbol{x}$, la fonction Jacobienne cesse d'être inversible sur $\boldsymbol{x}$ et l'on ne pourra plus appliquer les théorèmes de convergence précédents. La technique dite de bissection utilisée dans ce cas consiste à découper le pavé en sous-pavés $\boldsymbol{x}_i$ qui soit peuvent être éliminés immédiatement si $\boldsymbol{f}(\boldsymbol{x}_i)$ ne contient pas 0, soit peuvent se voir appliquer avec succès une méthode de Newton par intervalles, soit enfin seront découpés à leur tour.

Cela réclame de stocker les pavés en attente de traitement dans une file d'attente. De même, le résultat de l'algorithme sera une liste de pavés satisfaisant un critère d'arrêt à préciser et susceptibles de contenir un zéro de f. Le squelette de l'algorithme est le suivant :

Données : $\boldsymbol{f}$ une extension intervalle de f et $\boldsymbol{f}'$ une extension intervalle de f'
$\boldsymbol{x}$ un pavé de recherche
Résultat : Res une liste de pavés susceptibles de contenir un zéro de f
Initialisations :
$\mathcal{L} := \{\boldsymbol{x}\}$ la liste des pavés en attente de traitement
Res $:= \emptyset$
tant que $\mathcal{L} \neq \emptyset$ **faire**
sortir un pavé $\boldsymbol{x}$ de $\mathcal{L}$
$\boldsymbol{x}' := N(\boldsymbol{x}, \tilde{x})$ avec N un pas d'une méthode de Newton
si $\boldsymbol{x}' \neq \emptyset$ alors
si $\boldsymbol{x}'$ satisfait le critère d'arrêt alors
si $N(\boldsymbol{x}', \tilde{x}') \subset \boldsymbol{x}'$ alors « $\boldsymbol{x}'$ contient un zéro »
si $N(\boldsymbol{x}', \tilde{x}') \subset \text{int}(\boldsymbol{x}')$ alors « $\boldsymbol{x}'$ contient un unique zéro »
ranger $\boldsymbol{x}'$ dans Res
sinon
si $\boldsymbol{x}'$ est assez réduit par rapport à $\boldsymbol{x}$ alors
ranger $\boldsymbol{x}'$ dans $\mathcal{L}$
sinon
$(\boldsymbol{x}_1, \boldsymbol{x}_2) := \text{split}(\boldsymbol{x}')$ (bissection de $\boldsymbol{x}'$)
si $\boldsymbol{f}(\boldsymbol{x}_1) \ni 0$ alors ranger $\boldsymbol{x}_1$ dans $\mathcal{L}$
idem pour $\boldsymbol{x}_2$

Il reste à préciser le test d'arrêt, le test de réduction et la façon de découper $\boldsymbol{x}'$ en deux ou éventuellement plus.

Le test d'arrêt doit mesurer si $\boldsymbol{x}'$ est assez petit, si $\boldsymbol{f}(\boldsymbol{x}')$ est petit également et si une bissection peut améliorer le résultat trouvé [7]. Le test sur la taille de $\boldsymbol{x}'$ mesure la précision relative obtenue pour le zéro contenu dans $\boldsymbol{x}'$, il porte sur sa précision relative et s'écrit $\max_i w_r(\boldsymbol{x}'_i) \leq \varepsilon$ avec, pour chaque composante $\boldsymbol{x}'_i$ de $\boldsymbol{x}'$, $w_r(\boldsymbol{x}'_i) = w(\boldsymbol{x}'_i)/|\text{mid}(\boldsymbol{x}'_i)|$ si $\text{mid}(\boldsymbol{x})'_i \neq 0$ et $w_r(\boldsymbol{x}'_i) = w(\boldsymbol{x}'_i)$ sinon ; ce test remplace le test sur la stagnation des itérés utilisé pour l'algorithme de Newton flottant. Le seuil ε pour la taille de $\boldsymbol{x}'$ peut aller jusqu'à u la précision machine puisque des bissections répétées permettent d'atteindre cette valeur. Le test $w(\boldsymbol{f}(\boldsymbol{x}')) \leq \varepsilon'$ indique si le résidu $\boldsymbol{f}(\boldsymbol{x}')$ est assez proche de 0 : il s'agit d'un test sur la précision absolue du résidu. Le seuil ε' pourrait aller jusqu'à η le plus petit nombre flottant strictement positif, cependant l'accumulation d'erreurs d'arrondi et de surencadrements liés à l'arithmétique par intervalles interdisent d'atteindre cette valeur. La dernière partie du test d'arrêt détermine si une bissection peut permettre d'affiner cette valeur, selon les auteurs [7, 43], ce test compare $\boldsymbol{f}(\boldsymbol{x}')$ à $\boldsymbol{f}(\text{mid}(\boldsymbol{x}'))$ ou à $\boldsymbol{f}(\boldsymbol{x}_1)$ et $\boldsymbol{f}(\boldsymbol{x}_2)$ avec $(\boldsymbol{x}_1, \boldsymbol{x}_2) = \text{split}(\boldsymbol{x}')$: une bissection est inutile si $w(\boldsymbol{f}(\text{mid}(\boldsymbol{x}'))) \geq \nu w(\boldsymbol{f}(\boldsymbol{x}'))$ ou si $\max(w(\boldsymbol{f}(\boldsymbol{x}_1)), w(\boldsymbol{f}(\boldsymbol{x}_2))) \geq w(\boldsymbol{f}(\boldsymbol{x}'))$. La première solution impose de fixer un ν convenable, la seconde de calculer éventuellement inutilement $\boldsymbol{f}(\boldsymbol{x}_1)$ et $\boldsymbol{f}(\boldsymbol{x}_2)$.

Le test de réduction mesure si un pas de la méthode de Newton par intervalles $\boldsymbol{x}' := N(\boldsymbol{x}, \tilde{x})$ a réduit le pavé $\boldsymbol{x}$ ou non. En général ce test s'écrit $w(\boldsymbol{x}') \leq \alpha w(\boldsymbol{x})$ avec α un paramètre $\in]0;1]$ à fixer également ; dans le cas où $w(\boldsymbol{x}') > \alpha w(\boldsymbol{x})$, le pavé $\boldsymbol{x}'$ est insuffisamment réduit et il est coupé en deux avant d'être inséré dans la liste $\mathcal{L}$ des pavés en attente. Ceci permet d'assurer la terminaison de l'algorithme puisque tout pavé finira par avoir une largeur relative inférieure au seuil prescrit.

Enfin, la stratégie de bissection a fait l'objet de nombreuses études théoriques et expérimentales [28, 20, 61, 16]. L'idée la plus simple consiste à couper en deux le côté ayant la plus grande largeur afin d'assurer une décroissance de la largeur du pavé. On peut également découper le côté sur lequel la fonction varie le plus, c'est-à-dire le côté $\boldsymbol{x}_i$ pour lequel $w(\boldsymbol{f}'(\boldsymbol{x})_i).w(\boldsymbol{x}_i)$ est maximal, dans l'espoir d'éliminer rapidement une moitié. Enfin, dans le cas où des divisions par des intervalles contenant 0 ont créé deux sous-intervalles dans l'une des dimensions du pavé, la bissection peut se résumer à retourner les deux sous-pavés correspondant. Il faut cependant que le « trou » entre les deux sous-intervalles soit assez large pour être intéressant et également assez centré pour que les deux sous-pavés soient assez réduits, cf. figure 5.

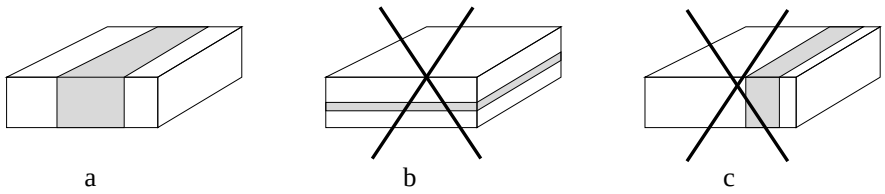

FIG. 5 – *Différents « trous » possibles après division par un intervalle contenant 0, seul le cas a est intéressant.*

Dans le cas d'un zéro multiple, $\boldsymbol{f}'(\boldsymbol{x})$ n'est pas régulière et seule la bissection permet de progresser dans l'algorithme, ce qui signifie que la complexité dans le pire cas est exponentielle en la dimension du problème et en les seuils. De plus, les comportements connus dans le cas de l'algorithme de Newton flottant se retrouvent avec la méthode de Newton par intervalles ; en particulier, la précision atteinte sur la racine est $u^{1/m}$ si m est la multiplicité de la racine : les bissections successives de $\boldsymbol{x}$ ne permettent pas d'éliminer une partie de $\boldsymbol{x}$ mais uniquement de satisfaire le critère d'arrêt sur la largeur de $\boldsymbol{x}$. En revanche, dans le cas de racines proches, la bissection conjuguée à une évaluation de f d'ordre élevé (avec un développement de Taylor à l'ordre 1 ou 2 — cf. [5] sur l'importance d'une bonne évaluation de la fonction dans la méthode de Newton par intervalles) permet de distinguer ces racines [43].

3.2.4 Techniques inspirées de la programmation logique sous contraintes

La résolution de systèmes non linéaires a également été étudiée par des spécialistes de la programmation logique sous contraintes. Ils ont adapté les notions utilisées en programmation logique au cas du calcul sur les intervalles [73, 11, 18]. Par exemple, une contrainte de la forme $c(x_1, \ldots, x_n)$ est dite consistante par arc avec l'ensemble E si et seulement si l'ensemble E est le plus petit ensemble de n-uplets satisfaisant la contrainte c ; plus formellement, cela se traduit par le fait que pour tout $x \in E$ et pour tout $i \in \{1, \ldots, n\}$, il existe $\exists x_j$ pour $j \neq i$ tels que $(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) \in E$ et $c(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$ est satisfaite. L'adaptation de cette notion au cas des intervalles a donné naissance aux notions de consistance par enveloppe convexe ou *hull consistency*, encore appelée consistance 2B, ou de consistance par pavé ou *box consistency*. Une contrainte de la forme $c(x_1, \ldots, x_n) = 0$ avec $x_i \in I\!R$ pour $1 \leq i \leq n$ est dite consistante par pavé avec le pavé $\boldsymbol{B} = \boldsymbol{B}_1 \times \ldots \boldsymbol{B}_n \in I\!I\!R^n$, $\boldsymbol{B}_i \in I\!I\!R$ pour $1 \leq i \leq n$ si et seulement si

$$\forall x_i \in \boldsymbol{B}_i,\ \exists x_j \in \boldsymbol{B}_j \text{ pour } j \neq i \ / \ c(x_1, \ldots x_{i-1}, x_i, x_{i+1}, \ldots x_n) = 0.$$

Autrement dit le pavé $\boldsymbol{B}$ est le pavé-enveloppe contenant l'ensemble pour lequel la contrainte est consistante par arc.

Une technique couramment utilisée en programmation logique pour résoudre un ensemble de contraintes, c'est-à-dire pour déterminer le pavé sur lequel elles sont toutes consistantes par pavé, est appelée *propagation de contraintes* et son adaptation à la résolution de systèmes de contraintes réelles est présentée ci-dessous. On suppose qu'il s'agit d'une contrainte donnée par un programme et dont la valeur de sortie est fixée. Ce programme est décomposé en une suite d'instructions élémentaires de la forme $x_k := x_i \diamond x_j$ ou $x_k := \varphi(x_i)$ où les opérations réciproques – au moins en calcul réel, même si ce n'est plus valable avec des intervalles – de $\diamond$ ($\diamond_g^{-1}$ et $\diamond_d^{-1}$ pour les réciproques à gauche et à droite) et de φ (à savoir φ^{-1}) sont connues. Par exemple si $\diamond$ est l'addition alors $\diamond^{-1}$ est la soustraction et si $\varphi = \sin$ alors $\varphi^{-1} = \arcsin$. Le programme de résolution s'écrit :

Initialisations : initialiser les valeurs de sortie aux valeurs désirées
tant que une amélioration a été apportée **faire**
forward propagation
évaluer dans l'ordre les instructions élémentaires :
$x_k := (x_i \diamond x_j) \cap x_k$ ou $x_k := \varphi(x_i) \cap x_k$
backward propagation
pour chaque instruction élémentaire dans l'ordre inverse
si cette instruction élémentaire est $x_k := x_i \diamond x_j$ alors
évaluer $x_i := (x_k \diamond_d^{-1} x_j) \cap x_i$ et $x_j := (x_i \diamond_g^{-1} x_k) \cap x_j$
sinon (elle est de la forme $x_k := \varphi(x_i)$)
évaluer $x_i := \varphi^{-1}(x_k) \cap x_i$

On s'arrête quand aucune variable intervalle (donnée ou intermédiaire) n'est modifiée. Cette technique permet de réduire efficacement les pavés de recherche initiaux et est utilisée notamment dans Numerica [74], Prolog IV [10] et Alias [49].

La méthode de Newton tout comme la méthode de propagation des contraintes sont des méthodes contractantes, qui s'appliquent donc particulièrement bien à l'arithmétique par intervalles. De plus, les théorèmes de point fixe (théorème de Brouwer, théorème de point fixe contractant) deviennent des théorèmes effectifs avec cette arithmétique, puisqu'il est facile de vérifier une inclusion telle que $\boldsymbol{f}(\boldsymbol{x}) \subset \boldsymbol{x}$ ou $\boldsymbol{f}(\boldsymbol{x}) \subset \text{int}(\boldsymbol{x})$.

3.3 Optimisation globale

Cette caractéristique de l'arithmétique par intervalles de permettre le calcul de l'image d'un ensemble par une fonction sera particulièrement exploitée dans ce paragraphe. En effet, elle assurera, grâce à l'information globale ainsi fournie, que l'algorithme d'optimisation détermine bien un optimum global et qu'il n'est pas piégé par un optimum local de moins bonne qualité.

Dans cette partie, nous allons tout d'abord présenter un algorithme pour l'optimisation globale sans contraintes d'une fonction continue dû à Hansen [28], puis le traitement des contraintes pour le problème de l'optimisation globale sous contraintes d'une fonction continue sera développé.

3.3.1 Optimisation globale sans contraintes

Par la suite et sans perte de généralité, le problème étudié sera un problème de minimisation, puisque maximiser une fonction f revient à minimiser $-f$. Il s'agit de déterminer f^* le minimum de la fonction et x^* le (ou les) point(s) également appelé(s) minimum[3] (minima) en lequel cette valeur est obtenue : soit $f : \mathbb{R}^n \to \mathbb{R}$ donnée, on cherche x^* et f^* tels que $f^* = f(x^*) = \min_{x \in \mathbb{R}^n} f(x)$. Les hypothèses de travail sont les suivantes :

- un pavé $\boldsymbol{x}_0$ contenant le minimum ou les minima x^* est connu ;
- ce minimum ou ces minima se trouvent à l'intérieur de $\boldsymbol{x}_0$ et non sur sa frontière ;
- la fonction f à minimiser est suffisamment régulière : l'algorithme présenté ci-dessous suppose que f est deux fois continûment dérivable ($f \in \mathcal{C}^2$) ; si f est seulement de classe $\mathcal{C}^1$ ou $\mathcal{C}^0$, on peut supprimer de l'algorithme les procédures utilisant les dérivées d'ordre 2 ou 1 sans perte de validité (il n'en va pas nécessairement de même pour l'efficacité...) ;
- une fonction d'inclusion $\boldsymbol{f}$ de f est disponible, ainsi qu'une fonction d'inclusion $\boldsymbol{G}$ de grad$f$ et qu'une fonction d'inclusion $\boldsymbol{H}$ de la Hessienne de f.

L'algorithme d'optimisation globale de Hansen est un algorithme de type *Branch and Bound* ou d'optimisation par séparation et évaluation pour reprendre la terminologie de [27].

3. L'anglais est une langue plus précise puisqu'on utilise le mot *minimum* pour la valeur minimale et le mot *minimizer* pour le point en lequel ce minimum est atteint.

On dispose d'une file de priorité notée $\mathcal{L}$ de pavés en attente de traitement, chaque pavé est soit rejeté s'il ne contient pas de minimum global, soit réduit si cela est possible[4], soit enfin découpé en plusieurs sous-pavés qui sont rangés dans la liste de priorité avant d'être traités à leur tour. Le résultat de cet algorithme est une liste de pavés $\boldsymbol{x}$ pour lesquels $\boldsymbol{f}(\boldsymbol{x})$ est proche du minimum trouvé.

Le squelette de l'algorithme est le suivant, les procédures le constituant seront détaillées par la suite.

Données : f et $\boldsymbol{f}$ une extension intervalle de f, $\boldsymbol{G}$ une extension intervalle de grad$f$, $\boldsymbol{H}$ une extension intervalle de la Hessienne de f

$\boldsymbol{x}_0$ le pavé de recherche

Résultats : $[\underline{f}; \overline{f}] \ni f^* = \min_{x \in \boldsymbol{x}_0} f(x)$

Res une liste de pavés $\boldsymbol{x}^*$ tels que $\boldsymbol{f}(\boldsymbol{x}^*) \cap [\underline{f}; \overline{f}] \neq \emptyset$

Initialisations :

$\mathcal{L} := \{\boldsymbol{x}_0\}$ la liste des pavés en attente de traitement

Res $:= \emptyset$

$[\underline{f}; \overline{f}] := \boldsymbol{f}(\boldsymbol{x}_0)$

tant que $\mathcal{L} \neq \emptyset$ **faire**

sortir un pavé $\boldsymbol{x}$ de $\mathcal{L}$

$\boldsymbol{x}' :=$ réduire $\boldsymbol{x}$

si $\boldsymbol{f}(\boldsymbol{x}') < \overline{f}$ alors

$\overline{f} := \max \boldsymbol{f}(\boldsymbol{x}')$

éliminer de Res les pavés $\boldsymbol{x}^*$ pour lesquels $\boldsymbol{f}(\boldsymbol{x}^*) > \overline{f}$

si $\boldsymbol{x}'$ satisfait le critère d'arrêt alors

ranger $\boldsymbol{x}'$ dans Res

sinon

$(\boldsymbol{x}_1, \boldsymbol{x}_2) :=$ split$(\boldsymbol{x}')$ (bissection de $\boldsymbol{x}'$)

si $\boldsymbol{x}_1$ ne peut pas être éliminé alors ranger $\boldsymbol{x}_1$ dans $\mathcal{L}$

idem pour $\boldsymbol{x}_2$

Fin : $\underline{f} := \min_{\boldsymbol{x} \in Res} \underline{\boldsymbol{f}(\boldsymbol{x})}$, $\overline{f} := \min_{\boldsymbol{x} \in Res} \overline{\boldsymbol{f}(\boldsymbol{x})}$.

Il reste à préciser les procédures de réduction, d'élimination et de bissection avant de préciser l'ordre choisi pour la file de priorité $\mathcal{L}$ et le test d'arrêt.

Procédure d'élimination

Trois tests peuvent être appliqués afin d'éliminer un pavé $\boldsymbol{x}$ qui ne contient pas de minimum global. Le premier de ces tests consiste simplement en une comparaison entre $\boldsymbol{f}(\boldsymbol{x})$ et le majorant courant $\overline{f}$ de f^* : si $\boldsymbol{f}(\boldsymbol{x}) > \overline{f}$ alors $\boldsymbol{x}$ ne peut pas contenir de minimum global. Le second critère est basé sur la propriété que le gradient de f s'annule en un minimum (ici on utilise le fait que le minimum n'est pas atteint sur la frontière de $\boldsymbol{x}_0$) : un pavé $\boldsymbol{x}$ est rejeté si $\boldsymbol{G}(\boldsymbol{x}) \not\ni 0$, il s'appelle test de monotonie. Enfin le dernier test, nommé test de convexité, repose sur le fait qu'en un minimum la fonction est localement convexe, ou de façon équivalente qu'en un minimum la Hessienne de f est symétrique positive définie (SPD). Comme cette propriété est délicate à tester, une caractérisation plus faible est utilisée, à savoir que la diagonale d'une matrice SPD est strictement positive. On l'utilise sous la forme suivante :

$$\begin{array}{l} \exists i \in \{1, \cdots, n\} \,/\, \boldsymbol{H}_{ii}(\boldsymbol{x}) < 0 \\ \Rightarrow \text{ Hessienne de } f \text{ en } \boldsymbol{x} \text{ non SPD} \\ \Leftrightarrow f \text{ non convexe en } \boldsymbol{x}. \end{array}$$

La figure 6 illustre ces trois tests d'élimination.

Procédure de réduction

La procédure de réduction vise à transformer le pavé $\boldsymbol{x}$ à traiter en un pavé $\boldsymbol{x}'$ inclus dans $\boldsymbol{x}$ et si possible plus petit. Elle est basée sur l'utilisation de développements de Taylor à l'ordre 1 et 2 et sur la méthode de Newton par intervalles vue au paragraphe 3.2.

Un développement de Taylor à l'ordre 1 s'écrit de la façon suivante en arithmétique par intervalles :

$$\text{soit } \check{x} = \text{mid}(\boldsymbol{x}),\ \forall y \in \boldsymbol{x},\ f(y) \in f(\check{x}) + \text{grad}f(\boldsymbol{x}).(y - \check{x}) \subset f(\check{x}) + \boldsymbol{G}(\boldsymbol{x}).(y - \check{x}).$$

4. On trouve parfois le terme *Branch and Prune* pour un algorithme qui réduit également le sous-domaine de recherche.

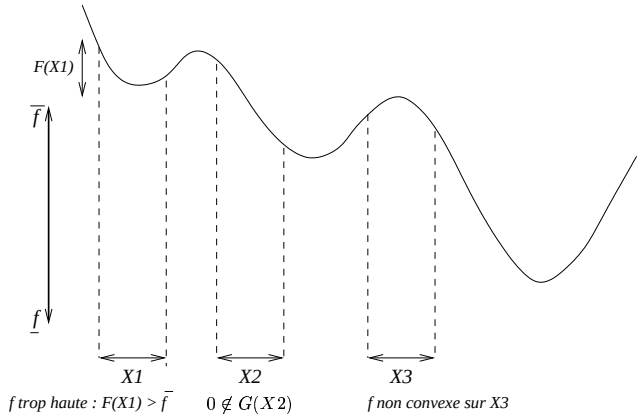

FIG. 6 – *Les trois tests d'élimination d'un pavé.*

On cherche $\boldsymbol{y} \subset \boldsymbol{x}$ pour lequel $\boldsymbol{f}(\boldsymbol{y}) \leq \overline{f}$, ou un surencadrement $\boldsymbol{y}_1$ de $\boldsymbol{y} \subset \boldsymbol{x}$ défini par

$$f(\check{x}) + \boldsymbol{G}(\boldsymbol{x}).(\boldsymbol{y}_1 - \check{x}) \leq \overline{f}.$$

On résout ces inégalités linéaires par une adaptation de la méthode de Gauss-Seidel au cas des inégalités et on remplace $\boldsymbol{x}$ par $\boldsymbol{y}_1$.

De la même façon, un développement de Taylor à l'ordre 2 conduit à déterminer $\boldsymbol{y}_2$ un surencadrement de $\boldsymbol{y}' \subset \boldsymbol{x}$ vérifiant

$$f(\check{x}) + \text{grad}f(\check{x}).(\boldsymbol{y}' - \check{x}) + (\boldsymbol{y}' - \check{x})^t.\boldsymbol{H}(\boldsymbol{x}).(\boldsymbol{y}' - \check{x}) \leq \overline{f}$$

par résolution (une composante après l'autre) d'inégalités quadratiques. On remplace alors $\boldsymbol{x}$ par $\boldsymbol{y}_2$.

Enfin, la dernière technique utilisée est courante en optimisation flottante : il s'agit d'appliquer un pas de la méthode de Newton par intervalles au gradient de la fonction f. Cette procédure permet même de prouver éventuellement l'existence et l'unicité de la solution dans le pavé étudié. Cette information peut être conservée par le stockage des pavés résultats en plusieurs listes, selon le nombre de minima contenus dans chaque pavé.

En pratique, les deux premières procédures sont rarement implantées et la réduction repose exclusivement sur la méthode de Newton.

Procédure de bissection

Après réduction éventuelle, le pavé $\boldsymbol{x}$ est découpé en deux sous-pavés dans l'algorithme indiqué, ou même en 4 ou 8 sous-pavés selon les auteurs [28, 61]. À nouveau, le choix de la direction à découper a été très étudié et on trouve par exemple le côté le plus long afin de diminuer la largeur des pavés, le côté ayant le plus grand trou créé par l'itération de Newton ou le côté sur lequel la fonction f varie le plus.

L'ordre dans lequel est rangée la file de priorité $\mathcal{L}$ influe beaucoup sur les performances, comme dans tout algorithme *Branch and Bound*. Hansen préconise une stratégie largeur d'abord pour assurer une subdivision uniforme des pavés à explorer, mais les besoins en mémoire sont importants. Les variantes de Moore-Skelboe et d'Ichida-Fujitsu [60] utilisent une stratégie meilleur d'abord, c'est-à-dire que les pavés $\boldsymbol{x}$ sont rangés dans l'ordre croissant des extrémités gauches de $\boldsymbol{f}(\boldsymbol{x})$. Enfin, Casado, García et Csendes [16] définissent un critère mesurant la distance relative de $\overline{f}$ au bord inférieur de $\boldsymbol{f}(\boldsymbol{x})$: si $\overline{f}$ est très proche de l'extrémité supérieure de $\boldsymbol{f}(\boldsymbol{x})$ qui est un surencadrement de $f(\boldsymbol{x})$, il est fort probable que $f(\boldsymbol{x})$ est supérieur à $\overline{f}$ et donc que $\boldsymbol{x}$ ne contienne pas de minimum global.

Enfin, le test d'arrêt porte sur les largeurs absolues du pavé $\boldsymbol{x}$ et de son image $\boldsymbol{f}(\boldsymbol{x})$, avec l'une d'entre elles qui doit être inférieure à un seuil donné [7]

Dans le cas où la fonction à optimiser est seulement continue par morceaux, un algorithme efficace d'optimisation n'utilisant pas les dérivées successives de f a été proposé et expérimenté par Jansson et Knüppel [32] : il s'agit d'une exploration de type *Branch and Bound* où la séparation est une bissection et où l'évaluation fait appel à un algorithme d'optimisation locale dû à Brent.

3.3.2 Optimisation globale sous contraintes

Le problème étudié dans ce paragraphe est de déterminer le minimum d'une fonction, ce minimum appartenant à un ensemble défini par des équations et des inéquations : soit $f : \mathbb{R}^n \to \mathbb{R}$ donnée, soient $c_i : \mathbb{R}^n \to \mathbb{R}$ pour $1 \leq i \leq l$ et soient $d_j : \mathbb{R}^n \to \mathbb{R}$ pour $1 \leq j \leq m$, on cherche x^* et f^* tels que

$$f^* = f(x^*) = \min_{x \in U} f(x) \text{ avec } U = \left\{ x \in \mathbb{R}^n \ / \ \begin{array}{l} c_i(x) = 0, 1 \leq i \leq l \\ d_j(x) \leq 0, 1 \leq j \leq m \end{array} \right\}.$$

Les hypothèses utilisées ici reprennent celles posées pour l'optimisation globale sans contraintes et incluent également le caractère $\mathcal{C}^2$ des fonctions c_i et d_j ainsi que l'existence de fonctions d'inclusion pour ces fonctions et leurs dérivées successives.

Le squelette de l'algorithme de Hansen pour l'optimisation globale sans contraintes peut être repris, mais le contenu des procédures d'élimination et de réduction change, ainsi éventuellement que les stratégies de bissection ou de gestion de la file de priorité ; en l'absence de résultats bien établis, ces deux derniers points ne seront pas abordés ici.

La procédure d'élimination élimine les pavés non réalisables qui sont les pavés $\boldsymbol{x}$ pour lesquels il existe un indice i tel que $c_i(\boldsymbol{x}) \not\ni 0$ ou $d_i(\boldsymbol{x}) > 0$, avant d'effectuer le test sur l'image « $\boldsymbol{f}(\boldsymbol{x}) > \overline{f}$? ».

La procédure de réduction commence par déterminer un sous-pavé $\boldsymbol{x}'$ de $\boldsymbol{x}$ sur lequel les contraintes sont satisfaites. Pour cela on peut appliquer la méthode de Newton par intervalles aux contraintes de type égalité (et ainsi prouver le cas échéant l'existence d'une solution réalisable). Pour les contraintes de type inégalité, on peut utiliser une technique analogue à celle employée pour résoudre $\boldsymbol{f}(\boldsymbol{y}) \leq \overline{f}$ au paragraphe précédent. Les techniques de propagation de contraintes peuvent évidemment être mises en œuvre. On peut également appliquer la méthode de Newton aux contraintes de type inégalité après les avoir transformées en contraintes de type égalité grâce à l'introduction de variables d'écart ; en effet, la contrainte $d_j(x) \leq 0$ équivaut aux contraintes $d_j(x) + \delta_j = 0$, $\delta_j \geq 0$. On remarque ici que le constat n'est plus valide selon lequel les contraintes de type égalité sont plus difficiles à gérer que celles de type inégalité parce qu'elles définissent un ensemble de volume nul — et donc que la « probabilité » de trouver un point réalisable est nulle. En effet la méthode de Newton par intervalles est très puissante pour ces contraintes de type égalité. En revanche, la gestion des contraintes actives de type inégalité n'est pas absolument satisfaisante... Les seules contraintes pour lesquelles on sait traiter les contraintes actives sont celles de la forme $a_i \leq x_i \leq b_i$: en effet on peut traiter séparément les trois problèmes $x_i = a_i$, $x_i = b_i$ (problèmes en dimension $n-1$) et $a_i \leq x_i \leq b_i$ où les bornes a_i et b_i servent uniquement à définir le pavé de recherche initial $\boldsymbol{x}_0$, mais sans qu'il soit envisagé que le minimum se trouve sur un bord, comme dans les hypothèses de l'algorithme d'optimisation sans contraintes. Cette technique de découplage en trois problèmes s'appelle *technique d'épluchage* ou *peeling technique* puisqu'on traite séparément l'intérieur et les bords de l'intervalle.

Enfin, la procédure de réduction peut appliquer la méthode de Newton par intervalles non à la fonction $\mathrm{grad} f$ puisque le gradient ne s'annule plus nécessairement en un minimum, mais au Lagrangien généralisé

$$\begin{pmatrix} \lambda_0 \mathrm{grad} f + \sum_{i=1}^{l} \lambda_i c_i + \sum_{j=1}^{m} \mu_j d_j \\ c_i \\ \mu_j d_j \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \text{ et } \mu_j \geq 0.$$

La présentation adoptée ici reprend celle de Hansen [28], qui recommande la formulation de Fritz-John (c'est-à-dire avec un multiplicateur de Lagrange λ_0) pour traiter correctement le cas de contraintes linéairement liées. Il explique également comment choisir une normalisation des multiplicateurs de Lagrange et comment déterminer des bornes pour ces coefficients. On peut alors appliquer la méthode de Newton par intervalles au système dont les $n+1+l+m$ variables sont les $(x_i)_{1 \leq i \leq n}$, λ_0, $(\lambda_i)_{1 \leq i \leq l}$, $(\mu_j)_{1 \leq j \leq m}$. Cela n'a apparemment pas encore été beaucoup mis en œuvre, mais la combinaison de la méthode de Newton appliquée au Lagrangien généralisé et de la technique de propagation de contraintes semble prometteuse.

La dernière modification à apporter à l'algorithme présenté pour l'optimisation sans contraintes concerne le majorant $\overline{f}$ de f^* : sa mise à jour ne peut avoir lieu que lorsque l'existence d'une solution réalisable est prouvée par la méthode de Newton, autrement on risquerait d'utiliser une valeur de $\overline{f}$ trop faible puisqu'obtenue sur un domaine trop grand et éventuellement non réalisable.

L'optimisation globale d'une fonction continue est un problème pour lequel l'arithmétique par intervalles, au contraire de l'arithmétique flottante, permet d'apporter une solution. Certaines briques de l'algorithme reprennent des techniques classiques en optimisation flottante, comme l'utilisation de la convexité, la méthode de Newton ou l'introduction de multiplicateurs de Lagrange généralisés, d'autres ne sont possibles qu'en arithmétique par intervalles, comme le test « $\boldsymbol{f}(\boldsymbol{x}) > \overline{f}$? ».

Toujours dans un souci de comparaison des méthodes flottantes et par intervalles, on peut constater que l'algorithme d'optimisation globale sous contraintes est une approche duale du problème d'optimisation, en ce sens qu'il approche le minimum par l'extérieur du domaine réalisable.

3.4 Liste d'autres problèmes bien résolus en arithmétique par intervalles

Le choix des problèmes et les algorithmes de résolution présentés dans ce qui précède est motivé d'une part par la quantité de publications sur ces sujets et d'autre part par les centres d'intérêt de l'auteur et est donc relativement subjectif. On peut mentionner d'autres problèmes traités avec succès par l'arithmétique par intervalles, comme la recherche des valeurs et vecteurs propres d'une matrice symétrique, l'inversion ensembliste ou l'intégration d'équations différentielles ordinaires avec conditions initiales.

Valeurs et vecteurs propres d'une matrice symétrique

Beaumont [9] détermine une caractérisation des éléments propres d'une matrice symétrique à l'aide d'un ensemble de programmes linéaires, en utilisant des majorations pour remplacer les termes quadratiques par des termes linéaires, et il calcule ensuite une boîte oblique contenant ces éléments propres.

Inversion ensembliste

Le problème de l'inversion ensembliste consiste, pour une fonction donnée $f : \mathbb{R}^m \to \mathbb{R}^n$ et un ensemble $Y \subset \mathbb{R}^n$, à déterminer des encadrements par l'intérieur $\underline{X}$ et par l'extérieur $\overline{X}$ de l'ensemble X tel que $f(X) = Y$. La méthode proposée par Jaulin [33] détermine $\underline{X}$ et $\overline{X}$ en utilisant l'arithmétique par intervalles : $\underline{X}$ est l'union de pavés de $\mathbb{R}^m$ tels que $\boldsymbol{f}(\underline{X}) \subset Y$ avec $\boldsymbol{f}$ une extension intervalle de f. L'encadrement extérieur $\overline{X}$ est l'union des pavés constituant $\underline{X}$ et des pavés dont l'image par $\boldsymbol{f}$ rencontre Y.

Intégration des équations différentielles ordinaires

L'une des premières applications de l'arithmétique par intervalles a été l'intégration des équations différentielles ordinaires avec conditions initiales [50]. Le problème de Cauchy est la recherche d'une solution x satisfaisant

$$\begin{cases} x' &= f(t,x), \\ x(t_0) &= x_0. \end{cases}$$

On prouve l'existence et l'unicité de la solution (sous certaines conditions) en montrant que l'opérateur de Picard

$$P \ : \ \varphi \in \mathcal{C}^1 \mapsto P(\varphi) : t \mapsto x_0 + \int_{t_0}^{t} f(u, \varphi(u)) du$$

est contractant. On peut donc itérer cet opérateur pour construire, en arithmétique par intervalles, la solution au problème de Cauchy [52].

Cette approche semble plus adaptée à l'arithmétique par intervalles que l'utilisation des méthodes de type Euler ou Runge-Kutta qui ont une fâcheuse tendance à donner des encadrements de plus en plus larges de $x(t)$ au fur et à mesure que t s'éloigne du point de départ t_0.

4 Logiciels et bibliothèques d'arithmétique par intervalles

L'arithmétique par intervalles a eu besoin de temps avant que les utilisateurs ne se convainquent de son intérêt. Une raison possible à cette méfiance est que les premières tentatives ont consisté à simplement remplacer le type « flottant » par le type « intervalle » sans rien modifier à l'algorithme, dans l'espoir d'estimer les erreurs d'arrondi. Cette démarche conduit à des encadrements exagérément larges à cause des problèmes d'effet enveloppant et de décorrélation des données. En revanche, elle n'exploite pas les spécificités de l'arithmétique par intervalles.

4.1 Compilateurs

IBM s'est cependant laissé persuader le premier de développer une extension du langage Fortran 77 nommée ACRITH implantant une arithmétique par intervalles, en collaboration avec Kulisch et son équipe à Karlsruhe en Allemagne.

Actuellement, Sun propose dans ses compilateurs Fortran 95 et C++ (Sun Forte) [71, 72] une extension des langages incluant le type intervalle, la surcharge des opérations arithmétiques et des fonctions élémentaires pour les intervalles, des opérations ensemblistes ($\cup$, $\cap$...) et spécifiques aux intervalles (mid, width...). Les bornes infinies sont autorisées.

4.2 Bibliothèques reposant sur l'arithmétique flottante

Le plus souvent, l'arithmétique par intervalles est réalisée sous la forme d'une bibliothèque. La suite du programme ACRITH a consisté en l'écriture de ACRITH-XSC, Pascal-XSC, C-XSC et C++-XSC [38], avec XSC pour *language eXtensions for Scientific Computation*, qui sont des bibliothèques implantant le type intervalle, les opérations arithmétiques et les fonctions élémentaires pour les objets de ce type, les types vecteurs et matrices d'intervalles. Elles réalisent en particulier le produit scalaire précis qui est calculé en arithmétique étendue, sur un nombre de bits suffisant pour permettre de couvrir toute la plage des exposants possibles en arithmétique flottante IEEE [41]. La version C++-XSC propose également des routines de résolution de systèmes d'équations linéaires ou non, polynomiales ou non. Elles sont disponibles à `http://www.uni-karlsruhe.de/~iam/html/language/xsc-sprachen.html`.

La bibliothèque fi_lib est une bibliothèque en C dont la particularité est de ne jamais changer le mode d'arrondi et de garantir que les résultats de tous les calculs, y compris les fonctions élémentaires, sont véritablement des encadrements des résultats cherchés. Pour cela, un ulp (*unit in the last place*) est ajouté ou retranché au résultat de chaque opération arithmétique ou algébrique. Les fonctions élémentaires quant à elles sont entièrement calculées par programme et un majorant des erreurs liées à l'approximation et au calcul flottant est ajouté ou soustrait des bornes du résultat. Cette bibliothèque est d'ailleurs utilisée par C-XSC pour les fonctions élémentaires. Elle est disponible à l'URL `http://www.uni-wuppertal.de/org/WRST/software.html`. La documentation [31] explique en détail (et en allemand) comment sont menés les calculs.

La bibliothèque BIAS/PROFIL est construite sur une partie en C, appelée BIAS pour *Basic Interval Algebra Subroutines* par analogie avec les BLAS (*Basic Linear Algebra Subroutines*), qui définit les types intervalle, vecteur d'intervalles et matrice d'intervalles ainsi que les opérations arithmétiques entre ces types. Les fonctions élémentaires sont les fonctions flottantes « corrigées » en ajoutant et retranchant un ulp aux extrémités des résultats...ce qui traduit un excès d'optimisme quant à la précision de ces fonctions (à ce sujet, voir par exemple [44]). Au-dessus de BLAS est défini PROFIL [39], écrit en C++ (hélas, un C++ ancien qui n'est plus compatible avec les nouveaux compilateurs). PROFIL surcharge les opérations arithmétiques pour simplifier l'écriture des programmes. De plus, il définit le type « fonctions différentiables » et permet de calculer, en mode *forward*, leurs différentielles (gradient et Hessienne) toujours en surchargeant les opérateurs et fonctions standard. On peut télécharger PROFIL/BIAS à partir de la page Web `http://www.ti3.tu-harburg.de/Software/PROFIL.html`.

Un des grands noms de l'arithmétique par intervalles, Baker Kearfott, propose également sa bibliothèque Intlib [3] écrite en Fortran 90 sur le Web, `ftp://interval.louisiana.edu/pub/interval_math/Fortran_90_software/`. Elle définit le type intervalle et surcharge les opérateurs et fonctions élémentaires pour qu'ils acceptent des intervalles comme arguments. Intlib propose surtout une représentation des fonctions définies par l'utilisateur qui permet de calculer la valeur d'une fonction en même temps que celle de ses dérivées et également de la différentier automatiquement, en mode *backward*. Le cas des branchements conditionnels est traité.

4.3 Bibliothèques basées sur l'arithmétique multi-précision ou exacte

On a vu au §2.2.2 que très souvent la largeur d'un résultat est proportionnelle à la largeur des arguments. Cela incite à proposer une arithmétique par intervalles dans laquelle la précision des extrémités peut être raffinée à volonté. En effet, l'arithmétique sous-jacente n'est pas nécessairement l'arithmétique flottante, elle

peut être une arithmétique en précision arbitraire. Pour cela, il faut que l'arithmétique multi-précision offre les arrondis vers l'extérieur, ou à tout le moins des résultats qui sont garantis être inférieurs ou supérieurs au résultat exact du calcul.

Or ce n'est pas le cas des arithmétiques multi-précision proposées par Mathematica ou Maple ; en particulier les fonctions élémentaires sont calculées sans aucune garantie sur l'erreur commise. Or les paquetages intpak [19] et intpakX [25] (encore un document en allemand !) supposent que l'erreur commise est inférieure à $0,6$ ulp, de même l'erreur est supposée être bornée par 1 ulp dans Mathematica [37]. De plus, d'autres bugs (tels que l'interprétation de π comme un nombre flottant par Maple, dès lors qu'il se trouve dans une expression flottante ou intervalle) interdisent à ces paquetages d'être réellement fiables.

La bibliothèque MPFI pour *Multiple Precision Floating-point Interval library*, écrite en C et bientôt en C++ et disponible à partir de ma page Web `http://www.ens-lyon.fr/~nrevol/nr_software.html`, repose sur la bibliothèque MPFR (*Multiple Precision Floating-point Reliable library*) disponible à l'adresse `http://www.mpfr.org/`. L'adaptation des algorithmes de la troisième partie de cet article est en cours afin de tirer parti de la précision multiple ; en particulier la précision doit être automatiquement adaptée aux besoins du calcul, pour ne pas calculer avec plus de chiffres que nécessaire. C'est possible pour les algorithmes itératifs comme Newton ou l'optimisation globale : en effet l'encadrement de la solution est raffiné au fur et à mesure du calcul et la précision requise augmente de la même façon.

On peut envisager comme arithmétique sous-jacente une arithmétique rationnelle, comme celle proposée par Arithmos [14]), qui utilise des rationnels de taille limitée (c'est-à-dire que les nombres de chiffres du numérateur et du dénominateur sont maintenus petits), ce qui évite de pâtir des problèmes d'arrondis tant que les résultats des calculs sont représentables par de tels rationnels et également de souffrir des problèmes d'explosion de la taille des nombres manipulés, classiques en calcul formel. Il est également prévu qu'une arithmétique multi-précision soit utilisée.

4.4 Logiciels de haut niveau utilisant l'arithmétique par intervalles

L'Atelier de QUAlité numérique pour la REalisation de Logiciels Scientifiques Aquarels [58], développé à l'IRISA, regroupe plusieurs outils logiciels destinés à estimer et contrôler la précision de logiciels scientifiques, tels qu'un logiciel de type PARANOIA qui teste les caractéristiques de l'arithmétique flottante, une arithmétique aléatoire basée sur la méthode CESTAC (voir [75, 17])... et une arithmétique par intervalles.

Plus récemment, le forum de réflexion (voir le site du forum technique `http://www.netlib.org/blas/blast-forum`) sur les BLAS, routines d'algèbre linéaire hautement optimisées qui servent de briques de base pour les programmes de calcul scientifique, met au point une proposition de BLAS par intervalles. Les routines BLAS « typiques » telles que les produits matrice-vecteur ou matrice-matrice, la résolution de systèmes linéaires triangulaires ou les axpy sont incluses dans cette proposition, ainsi que des opérations spécifiques aux intervalles, comme l'accès aux bornes d'un intervalle ou le test d'inclusion. Rump [66] de l'université de Harburg en Allemagne a déjà réalisé dans INTLAB l'intégration des intervalles en Matlab, qui est un outil de calcul matriciel interactif très simple d'utilisation, avec des possibilités graphiques étendues et de nombreuses « boîtes à outils » dédiées à des domaines d'application spécifiques. Pour cela, il profite au maximum des performances des BLAS flottantes sur lesquelles repose Matlab : par exemple il réalise le produit de matrices intervalles en n'utilisant que des produits de matrices flottantes, et ce sans changer de mode d'arrondi au cœur des boucles, ce qui empêcherait de pipeliner les calculs et ferait perdre l'efficacité des BLAS. L'écriture d'algorithmes basés sur les BLAS est rendue possible par la représentation des intervalles sous la forme centre — rayon, au prix d'une perte de précision. Les fonctions disponibles, outre le produit de matrices, sont la résolution de systèmes linéaires denses ou creux, éventuellement sur- ou sous-dimensionnés, ainsi que la résolution de systèmes non linéaires. INTLAB peut être téléchargé à l'adresse `http://www.ti3.tu-harburg.de/rump/intlab`.

Enfin, les algorithmes de résolution de contraintes par propagation vus au §3.2.4 sont implantés dans les logiciels commerciaux tels que Numerica [74] ou Prolog IV [10]. Le type intervalle est défini en Prolog IV, ainsi que l'union d'intervalles qui permet de représenter un ensemble de solutions distinctes d'un problème. Prolog IV permet de résoudre non seulement des contraintes purement logiques, mais également réelles, linéaires ou non, et renvoie les solutions sous forme d'unions d'intervalles.

5 Conclusion

L'arithmétique par intervalles constitue une bonne approche pour répondre à l'exigence de fiabilité des calculs. En effet, elle repose sur le principe que tout calcul retourne un encadrement garanti de son résultat. De plus, elle peut être implantée efficacement sur ordinateur et une panoplie d'algorithmes numériques ont été développés spécifiquement pour tirer parti de cette arithmétique. Il serait naïf de croire que les algorithmes utilisés en arithmétique flottante donnent des résultats probants sur des intervalles, le plus souvent les encadrements obtenus sont trop pessimistes. En revanche, des algorithmes conçus pour l'arithmétique par intervalles fournissent des informations et des résultats inaccessibles en flottant : rappelons le problème de l'optimisation globale d'une fonction continue pour lequel les algorithmes flottants retournent un optimum local. On peut également mentionner le problème de la résolution de systèmes non linéaires traité dans cet article : en utilisant l'arithmétique par intervalles, on peut non seulement déterminer toutes les solutions mais également prouver leur existence ou leur unicité, en utilisant les théorèmes de point fixe qui deviennent effectifs avec une telle arithmétique.

Toute médaille a son revers, celui de l'arithmétique par intervalles est de retourner des encadrements parfois trop larges des résultats. Nous croyons beaucoup aux possibilités offertes par la combinaison de l'arithmétique par intervalles et de l'arithmétique multi-précision, qui allie fiabilité et précision. Nos premières expériences semblent montrer que le surcoût dû à la précision multiple est compensé par l'économie de certains calculs inutiles.

Un dernier aspect qu'il est important de développer, ne serait-ce que pour convaincre par l'exemple de futurs utilisateurs, est la résolution de problèmes pratiques. En France, on peut citer l'intérêt croissant des automaticiens pour le calcul ensembliste en général et le calcul par intervalles en particulier ; le livre de Jaulin *et al.* [34] en fait foi.

Pour en savoir plus : la communauté d'arithmétique par intervalles a son site Web, assez complet : `http://www.cs.utep.edu/interval-comp/`.

Références

[1] O. Aberth. The solution of linear interval equations by a linear programming method. *Linear Algebra and its Applications*, 259:271–279, 1997.

[2] G. Alefeld and G. Mayer. Interval analysis: theory and applications. *Journal of Computational and Applied Mathematics*, 121:421–464, 2000.

[3] R. Baker Kearfott. A Fortran 90 environment for research and prototyping of enclosure algorithms for nonlinear equations and global optimization. *ACM TOMS*, 21(1):63–78, 1995.

[4] R. Baker Kearfott. *Rigorous global search: continuous problems.* Kluwer, 1996.

[5] R. Baker Kearfott. Empirical evaluation of innovations in interval branch and bound algorithms for nonlinear systems. *SIAM J. Sci. Comput.*, 18(2):574–594, 1997.

[6] R. Baker Kearfott, C. Hu, and M. Novoa. A review of preconditioners for the interval Gauss-Seidel method. *Interval Computations*, 1(1):59–85, 1991.

[7] R. Baker Kearfott and G.W. Walster. On stopping criteria in verified nonlinear systems or optimization algorithms. *ACM TOMS*, 26(3):373–389, 2000.

[8] E. Baumann. Optimal centered forms. *BIT*, pages 80–87, 1988.

[9] O. Beaumont. *Algorithmique pour les intervalles : comment obtenir un résultat sûr quand les données sont incertaines.* PhD thesis, IRISA, Université de Rennes 1, France, 1999.

[10] F. Benhamou, P. Bouvier, A. Colmerauer, H. Garetta, B. Giletta, J.-L. Massat, G.A. Narboni, S. N'Dong, R. Pasero, J.-F. Pique, Touraïvane, M. Van Caneghem, E. Vétillard, and J. Zhou. Manuel de Prolog IV. Technical report, PrologIA, 1996.

[11] F. Benhamou, F. Goualard, and L. Granvilliers. Revising hull and box consistency. In *International Conference on Logic Programming*, pages 230–244. The MIT Press, 1999.

[12] L. S. Blackford, A. Cleary, A. Petitet, R. C. Whaley, J. Demmel, I. Dhillon, H. Ren, K. Stanley, J. Dongarra, and S. Hammarling. Practical experience in the numerical dangers of heterogeneous computing. *ACM TOMS*, pages 133–147, 1997.

[13] C. Brezinski. *Projection methods for systems of equations.* North Holland, 1997.

[14] CANT Research Group. Arithmos: a reliable integrated computational environment. University of Antwerpen, Belgium, `http://win-www.uia.ac.be/u/cant/arithmos`, 2001.

[15] O. Caprani and K. Madsen. Mean value forms in interval analysis. *Computing*, 25:147–154, 1980.

[16] L. Casado, I. García, and T. Csendes. A new multisection technique in interval methods for global optimization. *Computing*, 65:263–269, 2000.

[17] J.-M. Chesneaux, S. Guilain, and J. Vignes. *La bibliothèque CADNA : présentation et utilisation.* `http://www-anp.lip6.fr/cadna`, 1996.

[18] H. Collavizza, F. Delobel, and M. Rueher. Comparing partial consistencies. *Reliable Computing*, 5(1):1–16, 1999.

[19] A.E. Connell and R.M. Corless. An experimental interval arithmetic package in Maple. In *Num. Analysis with Automatic Result Verification*, 1993.

[20] T. Csendes and D. Ratz. A review of subdivision direction selection in interval methods for global optimization. *Zeitschrift für Angewandte Mathematik und Mechanik*, 76(S1):319–322, 1996.

[21] M. Daumas and J.-M. Muller. *Qualité des calculs sur ordinateur.* Masson, 1997.

[22] M. Daumas, L. Rideau, and L. Théry. A generic library of floating-point numbers and its application to exact computing. In *14th International Conference on Theorem Proving in Higher Order Logics*, pages 169–184, Edinburgh, Scotland, 2001.

[23] J.W. Demmel. Trading off parallelism and numerical stability. In Marc S. Moonen, Gene H. Golub, and Bart L. De Moor, editors, *Linear Algebra for Large Scale and Real-Time Applications*, volume 232, pages 49–68. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993.

[24] A.A. Gaganov. Computational complexity of the range of the polynomial in several variables. *Cybernetics*, pages 418–425, 1985.

[25] I. Geulig and W. Kräemer. Intervallrechnung in Maple - Die Erweiterung intpakX zum Paket intpak der Share-Library. Technical Report 99/2, Universität Karlsruhe, 1999.

[26] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991.

[27] M. Gondran and M. Minoux. *Graphes et algorithmes.* Eyrolles, 1995.

[28] E. Hansen. *Global optimization using interval analysis.* Marcel Dekker, 1992.

[29] P. Hertling. A lower bound for range enclosure in interval arithmetic. In *Real Numbers and Computers 3*, 1998.

[30] N. Higham. *Accuracy and stability of numerical algorithms.* SIAM, 1996.

[31] W. Hofschuster and W. Krämer. *FI_LIB, eine schnelle und portable Funktionsbibliothek für reelle Argumente und reelle Intervalle im IEEE-double-Format.* Universität Karlsruhe, Preprint 98/7, 1998.

[32] C. Jansson and O. Knüeppel. Numerical results for a self-validating global optimization method. Technical Report 94.1, Technische Universität Hamburg-Harburg, 1994.

[33] L. Jaulin. *Solution globale et garantie de problèmes ensemblistes; application à l'estimation non linéaire et à la commande robuste.* PhD thesis, Université Paris-Sud Orsay, 1994.

[34] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied interval analysis.* Springer Verlag, 2001.

[35] W. Kahan. Lecture notes on the status of the IEEE Standard 754 for binary floating-point arithmetic. `http://www.cs.berkely.edu/~wkahan/ieee754status/ieee754.ps`, 1990.

[36] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[37] J. Keiper. Interval arithmetic in Mathematica. *Interval Computations*, (3), 1993.

[38] R. Klatte, U. Kulisch, C. Lawo, M. Rauch, and A. Wiethoff. *C-XSC a C++ class library for extended scientific computing.* Springer Verlag, 1993.

[39] O. Knueppel. PROFIL/BIAS - a fast interval library. *Computing*, 53(3-4):277–287, 1994.

[40] V. Kreinovich and J. Wolff von Gudenberg. An optimality criterion for arithmetic of complex sets. *Combinatorics*, 10(1):31–37, 2000.

[41] U. Kulisch and W. Miranker. The arithmetic of digital computer: a new approach. *SIAM Review*, 28(1):1–40, 1986.

[42] A.V. Lakeyev. Computational complexity of estimation of generalized sets of solutions for interval linear systems. In *SCAN*, Budapest, Hungary, pages 84–85, 1998.

[43] P. Langlois and N. Revol. Validating polynomial numerical computations with complementary automatic methods. Technical Report RR-4205, INRIA, `http://www.inria.fr/rrrt/rr-4205.html`, 2001.

[44] V. Lefèvre and J.-M. Muller. Worst cases for correct rounding of the elementary functions in double precision. In *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, Vail, Colorado, 2001. IEEE Computer Society Press, Los Alamitos, CA.

[45] X. Li, J. Demmel, D. Bailey, G. Henry, Y. Hida, J. Iskandar, W. Kahan, A. Kapur, M. Martin, T. Tung, and D.J. Yoo. Design, implementation and testing of extended and mixed precision BLAS. Technical Report CS-00-451, University of Tennessee (submitted to ACM TOMS), 2000.

[46] S. Markov. On directed interval arithmetic and its applications. *J. UCS*, 1(7):510–521, 1995.

[47] G. Mayer. Epsilon-inflation in verification algorithms. *Journal of Computational and Applied Mathematics*, 60:147–169, 1995.

[48] G. Mayer and J. Rohn. Feasibility of the preconditioned interval Gaussian algorithm. In *SCAN,Budapest, Hungary*, page 105, 1998.

[49] J.-P. Merlet. *ALIAS An Algorithms Library of Interval Analysis for equation Systems.* `http://www-sop.inria.fr/coprin/logiciels/ALIAS/ALIAS.html`, 2000.

[50] R. Moore. *Interval arithmetic and automatic error analysis in digital computing.* PhD thesis, Applied Math Statistics Lab., Report 25, Stanford, 1962.

[51] R.E. Moore. *Interval analysis.* Prentice Hall, 1966.

[52] R.E. Moore. *Methods and applications of interval analysis.* SIAM Studies in Applied Mathematics, 1979.

[53] A. Neumaier. *Interval methods for systems of equations.* Cambridge University Press, 1990.

[54] A. Neumaier. The wrapping effect, ellipsoid arithmetic, stability and confidence regions. *Computing Supplementum*, 9:175–190, 1993.

[55] A. Neumaier. A simple derivation of the Hansen-Bliek-Rohn-Ning-Kearfott enclosure for linear interval equations. `http://solon.cma.univie.ac.at/\~{}neum`, 1998.

[56] W. Oettli. On the solution set of a linear systems with inaccurate coefficients. *SIAM J. Numer. Anal.*, pages 115–118, 1965.

[57] W. Oettli and W. Pragger. Computability of approximate solution of linear equations with given error bounds for coefficients and right-hand sides. *Numer. Math*, 6:405–409, 1964.

[58] J.-C. Paoletti. Fortran Aquarels. RAP.TS.93.SEP.34, v.1-0, Simulog, 1993.

[59] D.M. Priest. Algorithms for arbitrary precision floating point arithmetic. In P. Kornerup and D. Matula, editors, *10th Symposium on Computer Arithmetic*, pages 132–144, Grenoble, France, 1991.

[60] H. Ratschek and J. Rokne. *New computer methods for global optimization.* Ellis Horwood Ltd, 1988.

[61] D. Ratz. Improved techniques for gap-treating and box-splitting in interval Newton Gauss-Seidel steps for global optimization with validation. *Zeitschrift für Angewandte Mathematik und Mechanik*, 76(S1):323–326, 1996.

[62] J. Rohn. Linear Interval Equations: Computing Sufficiently Accurate Enclosures is NP-Hard. Technical Report 621, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, Czech Republic, 1995.

[63] J. Rohn and V. Kreinovich. Computing exact componentwise bounds on solutions of linear systems with interval data is NP-hard. *SIAM J. Matrix Anal. Appl.*, 16(2):415–420, 1995.

[64] S. Rump. *Kleine Fehlerschranken bei Matrixproblemen.* PhD thesis, Universität Karlsruhe, 1980.

[65] S. Rump. *Topics in validated computations, J. Herzberger ed.*, chapter Verification methods for dense and sparse systems of equations. Elsevier, 1994.

[66] S. Rump. *Developments in Reliable Computing, T. Csendes ed.*, chapter INTLAB - Interval Laboratory, pages 77–104. Kluwer, 1999.

[67] S. Rump. Fast and parallel interval arithmetic. *BIT*, 39(3):534–554, 1999.

[68] L. Schwartz. *Analyse.* Hermann, 1970.

[69] G.W. Stewart. What is rounding error? `http://www.cs.umd.edu/~stewart`, in `/pub/misc`, 2000.

[70] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* Springer-Verlag, 1980.

[71] SUN Microsystems, Inc. *C++ Interval Arithmetic Programming Reference*, 2000.

[72] SUN Microsystems, Inc. *Interval Arithmetic Programming Reference - Sun Workshop 6 Fortran 95*, 2000.

[73] P. Van Hentenryck, D. Mcallester, and D. Kapur. Solving polynomial systems using a branch and prune approach. *SIAM J. Numer. Anal.*, 34(2):797–827, 1997.

[74] P. Van Hentenryck, L. Michel, and Y. Deville. *Numerica, a modeling language for global optimization.* MIT Press, 1997.

[75] J. Vignes. A stochastic arithmetic for reliable scientific computation. *Mathematics and Computers in Simulation*, 35:233–261, 1993.

[76] J. von zur Gathen and J. Gerhard. *Modern computer algebra.* Cambridge University Press, 1999.

[77] C. Weber and F. Ducros. Large-eddy and Reynolds averaged Navier-Stokes simulations of turbulent flow over an airfoil. *Int. J. Comput. Fluid Dyn.*, 13:327–355, 2000.

[78] M.A. Wolfe. Interval mathematics, algebraic equations and optimization. *Journal of Computational and Applied Mathematics*, 124:263–280, 2000.

[79] P. Zimmermann. Arithmétique en précision arbitraire. *Calculateurs Parallèles, Réseaux et Systèmes Répartis*, 13, 2001.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399