Introduction à l'arithmétique par intervalles

Nathalie Revol
INRIA, projet Arénaire
LIP (UMR CNRS/INRIA/ENSL/UCBL)
46, allée d'Italie, F-69364 Lyon cedex, France
Nathalie.Revol@ens-lyon.fr

École Jeunes Chercheurs en Algorithmique et Calcul Formel, Grenoble, 01-04-2004

Mots clés. Arithmétique par intervalles, calcul garanti, information globale, grossissement des résultats, itération contractante, optimisation globale, algorithme de Hansen, algorithme de Newton par intervalles, résolution de systèmes linéaires par intervalles, résolution de contraintes continues.

1 Historique succinct et subjectif

La naissance de l'arithmétique par intervalles n'est pas datée avec certitude : pour une large partie de la communauté, son père est Ramon Moore qui l'a mentionnée pour la première fois en 1962 dans [30] et qui l'a définie de façon très complète et publiée en 1966 dans [31]. Pour d'autres, on trouve déjà dans un article de T. Sunaga daté de 1958 [53] les fondements de l'arithmétique par intervalles. Dans [18], l'origine de l'arithmétique par intervalles est attribuée à Rosalind Cecil Young qui l'aurait proposée dans sa thèse de doctorat à l'Université de Cambridge en 1931 [56]. Il est très facile de se procurer ces références, grâce au site `http://www.cs.utep.edu/interval-comp/`, rubrique *Early papers*.

Que l'arithmétique par intervalles soit née en 1931, en 1958 ou en 1962, son enfance se prolonge jusqu'à la fin des années 1970 : à ses débuts en effet, cette arithmétique semble rester assez confidentielle. J'en ignore les raisons et ne peux que soulever des hypothèses. La première est que le calcul scientifique ne dispose pas encore d'assez de puissance pour pouvoir se permettre de perdre un facteur de vitesse d'au moins 4 et de mémoire d'au moins 2 en changeant d'arithmétique. La seconde est que l'arithmétique flottante n'est pas encore assez bien spécifiée (arrondis en particulier) pour qu'implanter l'arithmétique par intervalles à partir de l'arithmétique flottante disponible alors soit une tâche aisée.

À partir des années 1980, l'arithmétique par intervalles prend son essor, en Allemagne particulièrement, sous l'impulsion d'U. Kulisch à Karlsruhe. Il réussit à convaincre Nixdorf de développer un processeur spécifique, puis IBM de mettre au point un jeu d'instructions et un compilateur [20] intégrant l'arithmétique par intervalles. Il va également former un grand nombre d'étudiants qui ont maintenant essaimé dans toute l'Allemagne et contribuent à la maintenir dans le peloton de tête des pays intervallistes.

À la même époque, l'arithmétique flottante voit naître la norme IEEE-754 (voir le cours d'Arnaud Tisserand) qui spécifie en particulier les modes d'arrondis et facilite l'implantation de l'arithmétique par intervalles. Cependant, l'arithmétique par intervalles peine à convaincre des utilisateurs potentiels. Il me semble qu'elle n'a pas répondu à des espoirs, à vrai dire injustifiés, quant à son utilisation comme outil de validation numérique de calculs flottants : en effet, tout calcul par

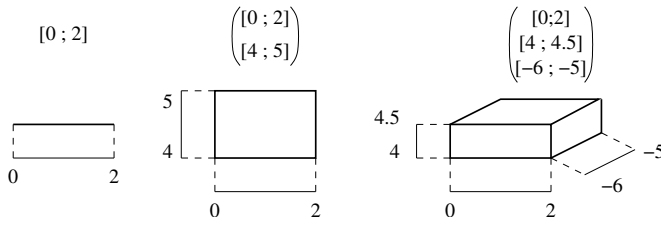

FIG. 1 – Des exemples de vecteur intervalle en dimensions 1, 2 et 3.

intervalle est un calcul *garanti*, c'est-à-dire que le résultat calculé est un intervalle garanti contenir la valeur ou l'ensemble de valeurs cherché. On s'imaginait alors qu'il suffisait de remplacer le type `float` ou `double` (ou `real`...) par le type `interval` dans un programme pour obtenir un résultat donnant un encadrement fin des erreurs d'arrondi, ce qui n'est pas le cas comme nous le verrons. Cet échec a desservi l'arithmétique par intervalles à ses débuts.

L'arithmétique par intervalles continue à se développer, mais avec des objectifs différents, depuis le début des années 1990. Son atout majeur, qui est de permettre de calculer sur des ensembles, est désormais mis à profit : c'est par exemple le seul outil déterministe (à ma connaissance) permettant de déterminer l'optimum global d'une fonction continue [36], de déterminer tous les zéros d'une fonction et de prouver en même temps leur existence et leur éventuelle unicité ou encore de déterminer l'image directe ou inverse d'un ensemble par une fonction [21].

2 Calcul par intervalles

2.1 Intervalles

En arithmétique par intervalles, on ne manipule plus des nombres, qui approchent plus ou moins fidèlement une valeur, mais des intervalles contenant cette valeur. Par exemple, on peut tenir compte d'une erreur de mesure en remplaçant une valeur mesurée x avec une incertitude ε par l'intervalle $[x-\varepsilon, x+\varepsilon]$. On peut également remplacer une valeur non exactement représentable, telle que π, par un intervalle la contenant ; si l'on dispose d'un ordinateur représentant les nombres en base 10 avec 3 chiffres, π sera remplacé par $[3.14, 3.15]$. Enfin, si l'on désire obtenir un résultat valide pour tout un ensemble de valeurs, on utilise un intervalle contenant ces valeurs. En effet, l'objectif de l'arithmétique par intervalles est de fournir des résultats qui contiennent à coup sûr la valeur ou l'ensemble cherché ; on parle alors de résultats *garantis* ou *validés*, ou encore *certifiés*.

Comme cela a été implicitement admis jusqu'à présent, les intervalles sont des sous-ensembles fermés connexes de $\mathbb{R}$. On notera $\mathbb{IR}$ l'ensemble des intervalles de $\mathbb{R}$. On peut les généraliser en plusieurs dimensions : un vecteur intervalle $\boldsymbol{x} \in \mathbb{IR}^n$ est un vecteur dont les n composantes sont des intervalles et une matrice intervalle $\boldsymbol{A} \in \mathbb{IR}^{m\times n}$ est une matrice dont les composantes sont des intervalles. Une représentation graphique d'un vecteur de $\mathbb{IR}$, $\mathbb{IR}^2$ et $\mathbb{IR}^3$ est donnée figure 1. Elle illustre le fait qu'un vecteur intervalle est un ensemble parallélépipédique de vecteurs aux côtés parallèles aux axes du repère ; cela justifie que par la suite on utilisera indifféremment les termes de vecteur intervalle, de pavé ou de boîte ou même d'intervalle.

Notations. Les objets intervalles seront désignés par des caractères gras : $\boldsymbol{x}$. On notera $\underline{x}$ le minimum de $\boldsymbol{x}$ et $\overline{x}$ son maximum, avec l'ordre partiel sur $\mathbb{R}^n$: $x \leq y$ ssi $x_i \leq y_i$ pour $1 \leq i \leq n$. On a alors $\boldsymbol{x} = [\underline{x}, \overline{x}]$. Enfin, $w(\boldsymbol{x})$ est la largeur de $\boldsymbol{x}$: $\overline{x} - \underline{x}$ (avec w pour *width*) ou encore son diamètre. Le centre $mid(\boldsymbol{x})$ et son rayon $rad(\boldsymbol{x})$ sont définis par $mid(\boldsymbol{x}) = (\underline{x} + \overline{x})/2$ et $rad(\boldsymbol{x}) = (\overline{x} - \underline{x})/2 = 1/2 w(\boldsymbol{x})$. On désignera par les adjectifs ¡¡ scalaire ¿¿ ou ¡¡ ponctuel ¿¿ un objet numérique usuel et on le confondra avec l'intervalle de largeur 0 ne contenant que cette valeur.

2.2 Calculs

Définition (abstraite)

Le résultat d'une opération entre deux intervalles : $\boldsymbol{x} \diamond \boldsymbol{y}$, resp. d'une fonction $f(\boldsymbol{z})$, est le plus petit intervalle au sens de l'inclusion (ou vecteur intervalle) contenant

$$\{x \diamond y \mid x \in \boldsymbol{x}, y \in \boldsymbol{y}\}, \text{ resp. } \{f(z) \mid z \in \boldsymbol{z}\},$$

c'est-à-dire le plus petit intervalle contenant tous les résultats possibles de l'opération appliquée à tous les éléments x de $\boldsymbol{x}$ et tous les éléments y de $\boldsymbol{y}$, resp. tous les résultats possibles de f appliquée à tous les éléments z de $\boldsymbol{z}$.

Opérations arithmétiques

Quand on applique la définition précédente aux opérations arithmétiques $=$, $-$, $\times$, ^{2}, $/$ ou $\sqrt{}$, on obtient les formules suivantes, plus utilisables en pratique que la définition abstraite :

$$\begin{array}{lcl}
[\underline{x}, \overline{x}] + [\underline{y}, \overline{y}] & = & [\underline{x} + \underline{y}, \overline{x} + \overline{y}] \\
[\underline{x}, \overline{x}] - [\underline{y}, \overline{y}] & = & [\underline{x} - \overline{y}, \overline{x} - \underline{y}] \\
[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] & = & [\min(\underline{x} \times \underline{y}, \underline{x} \times \overline{y}, \overline{x} \times \underline{y}, \overline{x} \times \overline{y}), \max(\underline{x} \times \underline{y}, \underline{x} \times \overline{y}, \overline{x} \times \underline{y}, \overline{x} \times \overline{y})] \\
[\underline{x}, \overline{x}]^2 & = & [\min(\underline{x}^2, \overline{x}^2), \max(\underline{x}^2, \overline{x}^2)] \text{ si } 0 \notin [\underline{x}, \overline{x}] \text{ et } [0, \max(\underline{x}^2, \overline{x}^2)] \text{ sinon} \\
1/\,[\underline{y}, \overline{y}] & = & [\min(1/\underline{y}, 1/\overline{y}), \max(1/\underline{y}, 1/\overline{y})] \text{ si } 0 \notin [\underline{y}, \overline{y}] \\
[\underline{x}, \overline{x}] \,/\, [\underline{y}, \overline{y}] & = & [\underline{x}, \overline{x}] \times (1/\,[\underline{y}, \overline{y}]) \text{ si } 0 \notin [\underline{y}, \overline{y}] \\
\sqrt{[\underline{x}, \overline{x}]} & = & [\sqrt{\underline{x}}, \sqrt{\overline{x}}] \text{ si } 0 \leq \underline{x}
\end{array}$$

On obtient ces formules en utilisant la monotonie (au moins partielle) de ces opérations.

Propriétés algébriques

On peut d'ores et déjà constater que les opérations définies ci-dessus ne présentent pas les propriétés algébriques de leurs contreparties ponctuelles. Tout d'abord, la soustraction n'est pas la réciproque de l'addition. Par exemple, si $\boldsymbol{x} = [2, 3]$, $\boldsymbol{x} - \boldsymbol{x} = [2, 3] - [2, 3] = [-1, 1] \neq 0$ même s'il le contient. En effet,

$$\boldsymbol{x} - \boldsymbol{x} = \{x - y \mid x \in \boldsymbol{x},\ y \in \boldsymbol{x}\} \supset \{x - x \mid x \in \boldsymbol{x}\} = \{0\}$$

et l'inclusion est stricte.
De la même façon, la division n'est pas la réciproque de la multiplication : si $\boldsymbol{x} = [2, 3]$, l'intervalle $\boldsymbol{x}/\boldsymbol{x} = [2, 3]/[2, 3] = [2/3, 3/2]$ n'est pas égal à 1 même s'il le contient.
De plus, la multiplication d'un intervalle par lui-même n'est pas égal à l'élévation au carré : si $\boldsymbol{x} = [-3, 2]$,

$$\boldsymbol{x} \times \boldsymbol{x} = [-3, 2] \times [-3, 2] = [-6, 9]$$

alors que

$$\boldsymbol{x}^2 = \{x^2 \mid x \in \boldsymbol{x}\} = [0, 9]\,.$$

Enfin, la multiplication n'est pas distributive par rapport à l'addition : si $\boldsymbol{x} = [-2, 3]$, $\boldsymbol{y} = [1, 4]$ et $\boldsymbol{z} = [-2, 1]$,

$$\begin{array}{lcl}
\boldsymbol{x} \times (\boldsymbol{y} + \boldsymbol{z}) & = & [-2, 3] \times ([1, 4] + [-2, 1]) \\
 & = & [-2, 3] \times [-1, 5] \\
 & = & [-10, 15] \\
\boldsymbol{x} \times \boldsymbol{y} + \boldsymbol{x} \times \boldsymbol{z} & = & [-2, 3] \times [1, 4] + [-2, 3] \times [-2, 1] \\
 & = & [-8, 12] + [-6, 4] \\
 & = & [-14, 16]
\end{array}$$

Comme l'illustre cet exemple, la multiplication est sous-distributive par rapport à l'addition, c'est-à-dire que

$$\boldsymbol{x} \times (\boldsymbol{y} + \boldsymbol{z}) \subset \boldsymbol{x} \times \boldsymbol{y} + \boldsymbol{x} \times \boldsymbol{z}.$$

La raison en est toujours que dans le premier cas on détermine $\{x \times (y+z) \mid x \in \boldsymbol{x},\ y \in \boldsymbol{y},\ z \in \boldsymbol{z}\}$ alors que dans le second on calcule $\{x \times y + x' \times z \mid x \in \boldsymbol{x},\ x' \in \boldsymbol{x},\ y \in \boldsymbol{y},\ z \in \boldsymbol{z}\}$, autrement dit on n'a pas identité de x et x'.

Fonctions élémentaires

On peut également définir des fonctions élémentaires (sin, exp, acoth...) prenant des intervalles pour argument, à l'aide de la définition abstraite ci-dessus. Pour les fonctions monotones telles que l'exponentielle ou l'arc-cotangente hyperbolique, on déduit facilement les formules permettant de les calculer :

$$\begin{array}{lcl} \exp([\underline{x},\, \overline{x}]) & = & [\exp \underline{x}, \exp \overline{x}] \text{ puisque exp est croissante,} \\ \operatorname{acoth}([\underline{x},\, \overline{x}]) & = & [\operatorname{acoth} \overline{x}, \operatorname{acoth} \underline{x}] \text{ si } [\underline{x},\, \overline{x}] \not\ni 0, \text{ puisque acoth est décroissante sur } \mathbb{R}^{+*} \text{ et } \mathbb{R}^{-*}. \end{array}$$

En revanche, il faut être plus soigneux pour les fonctions périodiques par exemple, mais il est possible d'établir des algorithmes de calcul pour ces fonctions, dès que l'on dispose de ces fonctions sur les réels. Par exemple, $\sin[\pi/3, \pi] = [0, 1]$.
Enfin, on ne sait définir les fonctions élémentaires que sur des intervalles inclus dans leur domaine de définition : on a vu ci-dessus que acoth n'était définie que pour des intervalles ne contenant pas 0, de la même manière le logarithme ne sera défini que pour des intervalles strictement positifs[1].

2.3 Évaluation d'une expression

Puisque l'on sait calculer le résultat d'une opération arithmétique ou d'une fonction élémentaire quand les variables prennent pour valeur des intervalles, on sait également calculer le résultat d'une expression mêlant opérations arithmétiques ou algébriques et fonctions élémentaires sur des intervalles. Voici simplement quelques exemples pour illustrer ce propos.
L'expression polynomiale $\boldsymbol{x}^3 - 2\boldsymbol{x}^2 + \boldsymbol{x} - 3$, avec $\boldsymbol{x} = [-5, 2]$, a pour résultat

$$\begin{array}{ll} & [-5,2]^3 - 2\,[-5,2]^2 + [-5,2] - 3 \\ = & [-125,8] - 2\,[0,25] + [-5,2] - 3 \\ = & [-125,8] - [0,50] + [-5,2] - 3 \\ = & [-183,7]\,. \end{array}$$

L'expression en plusieurs variables $\sin \boldsymbol{x} + 2\boldsymbol{x} \exp \boldsymbol{y} - \boldsymbol{y}^2\sqrt{\boldsymbol{z}}$ avec $\boldsymbol{x} = [-\pi, \pi/4]$, $\boldsymbol{y} = [-1, 1]$ et $\boldsymbol{z} = [1, 4]$ a pour résultat

$$\begin{array}{ll} & \sin\,[-\pi, \pi/4] + 2\,[-\pi, \pi/4] \times \exp\,[-1,1] - [-1,1]^2 \times \sqrt{[1,4]} \\ = & \left[-1, \sqrt{2}/2\right] + [-2\pi, \pi/2] \times [1/e, e] - [0,1] \times [1,2] \\ = & \left[-1, \sqrt{2}/2\right] + [-2\pi e, \pi e/2] - [0,2] \\ = & \left[-3 - 2\pi e, \sqrt{2}/2 + \pi e/2\right]. \end{array}$$

2.4 Implantations sur ordinateur

Comment implanter

Pour pouvoir implanter l'arithmétique par intervalles sur ordinateur, il faut être capable de retourner toujours un intervalle qui soit à la fois représentable en machine et contenant l'intervalle mathématique défini plus haut. Pour cela, il faut être capable de retourner un intervalle dont le minimum est inférieur ou égal au minimum de l'intervalle exact et dont le maximum est supérieur ou égal au maximum de l'intervalle exact[2]. C'est possible si on dispose d'arrondis dirigés pour le calcul ponctuel ; dans le cas où les intervalles sont représentés par leurs extrémités, on va

[1] C'est l'une des options possibles, l'autre étant de définir $f(\boldsymbol{x})$ comme $\{f(x) \mid x \in \boldsymbol{x} \cap \mathcal{D}_f\}$ où $\mathcal{D}_f$ est le domaine de définition, comme dans [21] ou [52].

[2] Ceci doit être vrai quelle que soit la représentation interne des intervalles, que ce soit par leurs extrémités, ou par leur centre et leur rayon ou toute autre représentation que vous pourrez imaginer.

arrondir vers le bas le résultat du calcul de la borne inférieure, et vers le haut le résultat du calcul de la borne supérieure. En particulier c'est le cas pour les opérations arithmétiques et algébriques définies par la norme IEEE-754 pour l'arithmétique flottante, qui est présentée dans le cours d'Arnaud Tisserand dans cette même école, mais hélas pas encore pour les fonctions élémentaires. On peut donc implanter les opérations arithmétiques en utilisant les modes d'arrondi disponibles (même s'ils sont plus ou moins facilement accessibles) et se retrousser les manches pour les fonctions élémentaires, ou utiliser une bibliothèque qui les calcule avec arrondis dirigés, comme Crlibm (cf. `http://perso.ens-lyon.fr/catherine.daramy/crlibm.html`) développée dans le projet Arénaire, ou MPFR (cf. `http://www.mpfr.org/`) du projet Spaces. Dans le cas où un intervalle est représenté par son centre et son rayon, il est possible, toujours grâce à l'arithmétique IEEE-754, de déterminer le centre du résultat, généralement en utilisant l'arrondi au plus près, puis le rayon du résultat en utilisant le mode d'arrondi vers le haut ainsi que des informations sur l'erreur commise en calculant le centre.

Quelques implantations

Mentionnons tout d'abord, pour en rester à la représentation par centre et rayon, le paquetage IntLab [49] utilisable en MatLab. Ce paquetage est développé et distribué par S. Rump. Cette représentation permet de calculer le centre de la matrice résultat (puisqu'en MatLab, tout est matrice) sans changer de mode d'arrondi pendant le calcul (on reste en arrondi au plus près) et les calculs matriciels optimisés de MatLab sont utilisés, puis de passer en arrondi vers le haut pour calculer le rayon, à nouveau en profitant des calculs matriciels rapides. En effet, tout changement de mode d'arrondi en cours de calcul ralentit le calcul (il faut vider le pipeline...). L'inconvénient de cette représentation est de conduire à des intervalles plus larges que ceux obtenus avec la représentation d'un intervalle par ses extrémités.

La grande majorité des implantations de l'arithmétique par intervalles utilisent en effet la représentation par extrémités. Elles utilisent très souvent aussi l'arithmétique flottante simple ou double précision disponible sur les processeurs. Séparons ces implantations en trois grandes catégories : compilateurs, bibliothèques et logiciels de plus haut niveau.

Du côté des compilateurs, IBM s'est laissé convaincre le premier, dans les années 1980, de développer une extension du langage Fortran 77, nommée ACRITH [20] et du compilateur qui allait avec. Cependant, les débuts de l'arithmétique par intervalles ayant été difficiles, IBM a renoncé. Actuellement, Sun propose dans ses compilateurs Sun Forte pour Fortran 95 [52] et C++ [51] une extension à l'arithmétique par intervalles : type intervalle, opérations arithmétiques, fonctions élémentaires, opérations ensemblistes telles que l'union ou l'intersection.

Les bibliothèques sont plus nombreuses. Les bibliothèques XSC, pour *language eXtension for Scientific Computing*, à savoir Pascal-XSC, C-XSC et C++-XC [24], ont pris la relève de ACRITH pour ces langages : type intervalle, opérations arithmétiques, fonctions élémentaires, vecteurs et matrices. La bibliothèque BIAS/PROFIL [26] comporte une couche basse en C, implantant les types simple, vecteur et matrice intervalles ainsi que les opérations arithmétiques sur ces types ; cette couche basse s'appelle BIAS pour *Basic Interval Algebra Subroutines*, par analogie avec les BLAS : *Basic Linear Algebra Subroutines*. Les fonctions élémentaires de BIAS ne sont pas garanties correctes. La couche haute, PROFIL [25], est écrite en C++ et contient des mécanismes de différentiation automatique : calcul des gradients et des Hessiennes en mode *forward*. Malheureusement cette bibliothèque est un peu ancienne et le C++ utilisé ne compile plus.

La bibliothèque fi_lib [27] est une bibliothèque en C dont les deux particularités sont de ne jamais changer le mode d'arrondi (et de fonctionner correctement pour tous les modes d'arrondi IEEE-754, sans savoir lequel est en usage) et de calculer correctement les fonctions élémentaires : celles-ci ont été complètement écrites et tiennent compte des erreurs d'approximation de ces fonctions et des erreurs d'arrondi. Enfin, un des grands noms de l'arithmétique par intervalles, B. Kearfott, propose également la bibliothèque intlib, écrite en Fortran 90. La bibliothèque intlib

met en place un mécanisme de différentiation automatique en mode *backward* et la gestion des branchements conditionnels. Grâce à intlib, B. Kearfott a développé et implanté de nombreux algorithmes et les a rendus efficaces en pratique, par ce constant retour de la pratique sur les développements théoriques [1].

On peut également trouver l'arithmétique par intervalles dans des logiciels non dédiés à cette arithmétique, comme Aquarels (Atelier de QUAlité numérique pour la REalisation de Logiciels Scientifiques) [35] développé il y a quelques années à l'IRISA, Prolog IV [7] qui utilise les intervalles pour la résolution de contraintes numériques, Numerica [55] qui était entièrement dédié à la résolution garantie de contraintes numériques (sa commercialisation a cessé), ou Alias [12], un solveur basé sur l'arithmétique par intervalles, dont le développement est encore en cours et qui ne cesse de s'enrichir de nouvelles méthodes.

Pour terminer, citons des paquetages ou bibliothèques d'arithmétique par intervalles en précision arbitraire, c'est-à-dire qui représentent les intervalles à l'aide de nombres rationnels exacts ou flottants en précision quelconque. Des paquetages utilisant les flottants en précision arbitraire existent en Maple : intpakX [15] et en Mathematica [23]. Cependant, les résultats calculés ont beau avoir beaucoup de chiffres, ils ne sont pas garantis puisque l'arithmétique flottante de ces deux systèmes est mal spécifiée. Arithmos [9] propose une représentation à l'aide de rationnels (exacts). Enfin MPFI [41] est une bibliothèque en C/C++ qui utilise les flottants en précision arbitraire de MPFR [16] pour représenter les intervalles et calculer des résultats garantis et précis.

3 Forces et faiblesses de l'arithmétique par intervalles

3.1 Force : calcul sur des ensembles

L'avantage majeur de l'arithmétique par intervalles est qu'elle permet de calculer avec des ensembles. C'est cette force qui a permis de surmonter la mauvaise impression laissée par des débuts peu concluants (estimation grossière des erreurs d'arrondi), de repartir vers des utilisations adaptées et de faire l'effort de développer des algorithmes qui tiennent compte des avantages et difficultés de l'arithmétique par intervalles.

Si l'on cherche à prouver qu'une valeur donnée n'est jamais atteinte – dans le cas où cette valeur correspond à un comportement dangereux ou à une situation incontrôlable par exemple, voir figure 2 – et que l'évaluation de l'expression sur un intervalle de valeurs d'entrées renvoie un résultat ne contenant pas cette valeur, on a la preuve cherchée. On peut par exemple chercher à déterminer si une sous-expresssion apparaissant au dénominateur d'une expression donnée peut s'annuler. Si l'évaluation de cette sous-expression sur (un intervalle contenant) l'ensemble des valeurs d'entrée considérées renvoie un intervalle ne contenant pas 0, on peut lancer le calcul, ponctuel ou par intervalles, sans craindre de division par 0. On peut également chercher si une zone dangereuse peut être atteinte. Là encore, si le résultat du calcul a une intersection vide avec ladite zone, cette zone est prouvée inaccessible.

L'arithmétique par intervalles, en permettant le calcul sur des ensembles, rend effectifs des théorèmes tels que le théorème de Brouwer/Schauder, sous-jacent dans les résultats du §5.3. Cela signifie qu'en calculant par intervalles, on peut obtenir des preuves, au sens mathématique du terme, de certaines propriétés. On a déjà vu des preuves d'absence de solution, on peut également obtenir des preuves d'existence seule ou d'existence et unicité de solutions, cf. §5.3.

3.2 Force : informations globales, optimisation globale

Savoir calculer sur des ensembles, même si on est restreint à des ensembles de forme particulière, permet également d'obtenir des informations globales précieuses pour l'optimisation globale d'une

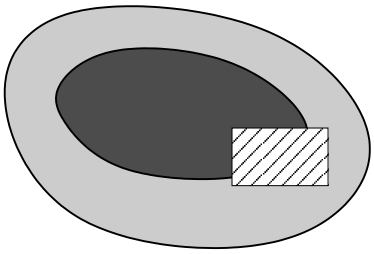

FIG. 2 – La zone gris sombre est la zone de comportement ¡¡ sûr ¿¿ et la zone gris clair est la zone de comportement moins sûr mais encore contrôlable. En dehors, le système n'est plus sûr ni contrôlable. La zone calculée (rectangle hachuré) ayant une intersection vide avec la zone non contrôlable, le système n'est pas dangereux.

fonction. Supposons que l'on cherche à déterminer le minimum global x^* d'une fonction f de $\mathbb{R}^n$ dans $\mathbb{R}$ assez régulière[3] (au moins continue et si possible de classe $\mathcal{C}^2$) sur un pavé $\boldsymbol{x}_0$, que l'on peut supposer être fourni par l'utilisateur. On suppose aussi que l'on dispose d'une expression de f, afin de pouvoir l'évaluer sur des intervalles. On supposera de plus qu'il s'agit d'un point stationnaire, c'est-à-dire d'un minimum sur lequel le gradient de f s'annule. Autrement dit, on cherche un minimum à l'intérieur de $\boldsymbol{x}_0$, ou encore on ne cherche pas à minimiser une fonction telle que $f(x) = x$ sur un intervalle quelconque.

Voici un premier algorithme, élémentaire, pour déterminer un petit intervalle contenant x^* :

Données : f la fonction à minimiser et $\boldsymbol{x}_0$ l'intervalle dans lequel on cherche le minimum
ε seuil de largeur des intervalles résultats

Résultats : $[\underline{f};\, \overline{f}] \ni f^* = \min_{x \in \boldsymbol{x}_0} f(x)$ // un encadrement de la valeur minimale de f sur $\boldsymbol{x}_0$
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 $:= \emptyset$
$[\underline{f};\, \overline{f}] := f(\boldsymbol{x}_0)$
évaluer f en un ou plusieurs points de $\boldsymbol{x}_0$, par exemple $\overline{f} := f(mid(\boldsymbol{x}_0))$
// on a maintenant un encadrement plus précis de $f(x^*)$ par $[\underline{f},\, \overline{f}]$

tant que $\mathcal{L} \neq \emptyset$ **faire**
sortir un pavé $\boldsymbol{x}$ de $\mathcal{L}$
couper $\boldsymbol{x}$ en deux : $\boldsymbol{x}_1$ et $\boldsymbol{x}_2$
évaluer $f(\boldsymbol{x}_1)$: $[\underline{f_1},\, \overline{f_1}]$
mettre à jour $\overline{f}$: si $\overline{f_1} \leq \overline{f}$ alors $\overline{f} := \overline{f_1}$
si $\underline{f_1} > \overline{f}$ alors
jeter $\boldsymbol{x}_1$ // $\boldsymbol{x}_1$ ne peut pas contenir le minimum x^*
sinon si $w(\boldsymbol{x}_1) \leq \varepsilon$ alors ranger $\boldsymbol{x}_1$ dans Res
sinon ranger $\boldsymbol{x}_1$ dans $\mathcal{L}$
idem avec $\boldsymbol{x_2}$

Fin :
mettre à jour l'encadrement de $f(x^*) = [\underline{f},\, \overline{f}] : \underline{f} = \min_{y=f(\boldsymbol{x}),x \in Res} \underline{y},\ \overline{f} = \max_{y=f(\boldsymbol{x}),x \in Res} \overline{y}$
éliminer les $\boldsymbol{x}$ de Res tels que $\underline{y} > \overline{f}$ avec $[\underline{y},\, \overline{y}] = f(\boldsymbol{x})$

Cet algorithme renvoie une liste d'intervalles, Res, qui contient tous les intervalles susceptibles de contenir un minimum global : sur chacun de ces intervalles, f prend des valeurs proches de l'optimum. On peut noter une caractéristique de l'arithmétique par intervalles : cet algorithme

[3]Dans ce qui suit, on supposera f de classe $\mathcal{C}^2$; si ce n'est pas le cas, il suffit de supprimer, dans les différents algorithmes, les procédures qui nécessitent une telle régularité.

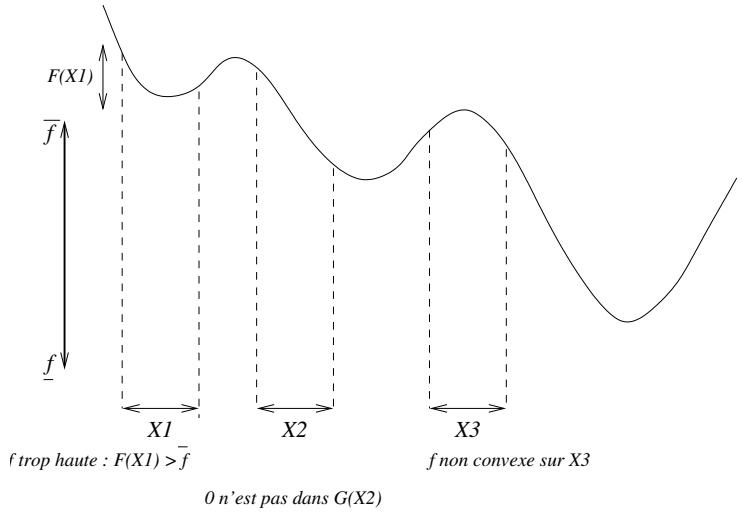

FIG. 3 – Critères permettant de rejeter un intervalle qui ne contient pas le minimum.

garantit de n'oublier aucun minimum global. Cet algorithme est aussi le seul envisageable si f est de classe $\mathcal{C}^0$ uniquement.

Si f est plus régulière, on peut appliquer différentes stratégies pour tenter d'éliminer rapidement ou de réduire un pavé. Par réduire $x = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, on entend diminuer au moins l'une des largeurs $w(\boldsymbol{x}_i)$ pour i entre 1 et n.
Les critères de rejet d'un pavé $\boldsymbol{x}$ sont les suivants :

- si $f(\boldsymbol{x}) = [\underline{y}, \overline{y}]$ est trop élevé : $\underline{y} > \overline{f}$ (c'est le cas de $X1$ sur la figure 3) ;
- si grad $f(\boldsymbol{x}) \not\ni 0$: $\boldsymbol{x}$ ne contient pas de point stationnaire (c'est le cas de $X2$ sur la figure 3) ;
- si Hess $f(\boldsymbol{x})$, la Hessienne de f sur $\boldsymbol{x}$, ne contient aucune matrice symétrique positive définie (c'est le cas de $X3$ sur la figure 3) : dans ce cas f n'est pas localement convexe sur $\boldsymbol{x}$; ceci est difficile à tester, en pratique on teste uniquement si Hess $f(\boldsymbol{x})$ contient une matrice à diagonale positive, ce qui est moins souvent vérifié mais plus facile à tester.

Dans tous les cas mentionnés ci-dessus, le pavé $\boldsymbol{x}$ ne peut pas contenir de minimum global et on cesse donc de l'examiner pour passer au prochain pavé à traiter. Si un intervalle $\boldsymbol{x}$ est encore en lice après ces trois tests on tente de le réduire avant de le remettre en attente de traitement et de le couper en deux.

Une première possibilité est de limiter la recherche au sous-pavé $\boldsymbol{x}'$ de $\boldsymbol{x}$ sur lequel $f(\boldsymbol{x}') \leq \overline{f}$: pour cela on peut soit utiliser un développement de Taylor (cf. §4) d'ordre 1 ou 2 de f et résoudre une inégalité affine ou quadratique, soit appliquer les techniques de résolution de contraintes (propagation et rétro-propagation) présentées au §7. En pratique, ces deux techniques sont rarement mises en œuvre.

Une seconde possibilité consiste à rechercher les zéros du gradient de f sur $\boldsymbol{x}$: on applique une itération ou un petit nombre d'itérations de l'algorithme de Newton par intervalles qui sera expliqué au §5.

L'algorithme d'optimisation globale, dû à Hansen [17], s'écrit donc.

Données :

f et $\boldsymbol{f}$ une extension intervalle de f (cf. §4),

$\boldsymbol{G}$ 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 $\boldsymbol{x}$:
résoudre $f(\boldsymbol{x}') \leq \overline{f}$ grâce à un développement de Taylor d'ordre 1
résoudre $f(\boldsymbol{x}') \leq \overline{f}$ grâce à un développement de Taylor d'ordre 2
appliquer quelques itérations de Newton à $\boldsymbol{G}$
couper $\boldsymbol{x}$ en deux : $\boldsymbol{x}_1$ et $\boldsymbol{x}_2$
évaluer $f(\boldsymbol{x}_1) : [\underline{f_1},\, \overline{f_1}]$
mettre à jour $\overline{f}$: si $\overline{f_1} \leq \overline{f}$ alors $\overline{f} := \overline{f_1}$
si $\underline{f_1} > \overline{f}$ ou si $\boldsymbol{G}(\boldsymbol{x}_1) \not\ni 0$ ou si $\boldsymbol{H}(\boldsymbol{x}_1)$ ne vérifie pas la condition de convexité alors
jeter $\boldsymbol{x}_1$ // $\boldsymbol{x}_1$ ne peut pas contenir le minimum x^*
sinon si $w(\boldsymbol{x}_1) \leq \varepsilon$ alors
ranger $\boldsymbol{x}_1$ dans Res
sinon
ranger $\boldsymbol{x}_1$ dans $\mathcal{L}$
idem avec $\boldsymbol{x_2}$
Fin :
mettre à jour l'encadrement de $f(x^*) = [\underline{f},\, \overline{f}]$:
$\underline{f} = \min_{y=f(\boldsymbol{x}), x \in Res} \underline{y},\ \overline{f} = \max_{y=f(\boldsymbol{x}), x \in Res} \overline{y}$
éliminer les $\boldsymbol{x}$ de Res tels que $\underline{y} > \overline{f}$ avec $[\underline{y},\, \overline{y}] = f(\boldsymbol{x})$

3.3 Force : informations globales, optimisation globale sous contraintes

Fréquemment, les problèmes d'optimisation globale sont des problèmes sous contraintes : on cherche le minimum d'une fonction f qui vérifie en outre un certain nombre d'équations et inéquations. Mathématiquement, le problème s'écrit :

$$(\mathcal{P}_c) \begin{cases} \min_x f(x) & & & & \\ \text{contraintes :} & p_i(x) & \leq & 0 & 1 \leq i \leq m \\ & q_j(x) & = & 0 & 1 \leq j \leq r \end{cases}$$

La difficulté est maintenant concentrée en deux points : déterminer des points qui satisfont les contraintes et trouver leur minimum x^*, qui lui ne vérifie plus grad $f(x^*) = 0$ ni f localement convexe en x^*.

Deux approches (au moins) sont possibles et complémentaires. La première est classique en mathématiques et analyse numérique : elle consiste à transformer $\mathcal{P}_c$ en un problème d'optimisation globale sans contraintes. Pour cela, les contraintes sont intégrées à la fonction à minimiser, sous forme de pénalités, ou alors en remplaçant dans Newton la fonction grad f par λ_0 grad $f + \sum_i \lambda_i$ grad $p_i + \sum_j \mu_j$ grad q_j où les λ_i et μ_j sont appelés multiplicateurs de Lagrange. Ces nouvelles variables doivent vérifier certaines conditions appelées conditions de Kuhn et Tucker si $\lambda_0 = 1$ et de Fritz - John dans le cas général. Ces conditions sont des égalités à 0 et peuvent dont être traitées par l'algorithme de Newton. Nous renvoyons à [17] pour une description plus précise et une discussion plus détaillée de cette méthode, qui inclut la détermination d'un pavé contenant les paramètres λ_i et μ_j, qui sera fourni en entrée de l'algorithme de Newton.

La seconde approche utilise les contraintes pour rejeter ou réduire le pavé de recherche à chaque étape. Un pavé $\boldsymbol{x}$ sera rejeté s'il ne satisfait pas les contraintes, par exemple s'il existe un indice i tel que $p_i(\boldsymbol{x}) > 0$ ou $q_i(\boldsymbol{x}) \not\ni 0$. Si ce pavé $\boldsymbol{x}$ n'est pas rejeté, on tentera d'éliminer de $\boldsymbol{x}$ les parties ne satisfaisant pas les contraintes. Pour cela, les techniques présentées au §7 ou dans [1] peuvent être mises en œuvre.

On peut n'utiliser qu'une seule de ces approches : dans le second cas, on supprime les appels à Newton pour résoudre grad $f = 0$ et le test de convexité. On peut aussi combiner les deux approches pour réduire à chaque étape le pavé courant autant que possible. Par exemple, un algorithme d'optimisation globale sous contraintes peut s'écrire de la façon suivante.

Données : f et $\boldsymbol{f}$ une extension intervalle de f,
$\boldsymbol{x}_0$ le pavé de recherche,
$\boldsymbol{p}_i$, $1 \leq i \leq m$, des extensions intervalles des p_i définissant les contraintes de type inégalité,
$\boldsymbol{q}_j$, $1 \leq j \leq r$, des extensions intervalles des q_j définissant les contraintes d'égalité à 0
Résultats : $[\underline{f}; \overline{f}] \ni f^* = \min_{x \in \boldsymbol{x}_0} f(x)$
Res une liste de pavés $\boldsymbol{x}^*$ tels que $\boldsymbol{f}(\boldsymbol{x}^*) \cap [\underline{f}; \overline{f}] \neq \emptyset$
pour certains éléments de Res, un certificat d'existence d'un point satisfaisant les contraintes
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 $\boldsymbol{x}$ // en utilisant les contraintes aussi : cf. ci-dessous
résoudre $\boldsymbol{f}(\boldsymbol{x}') \leq \overline{f}$
appliquer quelques pas de Newton à λ_0 grad $\boldsymbol{f} + \sum_i \lambda_i$ grad $\boldsymbol{p}_i + \sum_j$ grad $\boldsymbol{q}_j$
appliquer quelques pas de l'algorithme de propagation et rétro-propagation
s'il est prouvé que $\boldsymbol{x}'$ contient un point satisfaisant les contraintes et 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 $\boldsymbol{f}(\boldsymbol{x}_1) > \overline{f}$ ou s'il existe i tel que $\boldsymbol{p}_i(\boldsymbol{x}') > 0$ ou $\boldsymbol{q}_i(\boldsymbol{x}') \not\ni 0$ alors
éliminer $\boldsymbol{x}_1$
sinon
ranger $\boldsymbol{x}_1$ dans $\mathcal{L}$
idem pour $\boldsymbol{x}_2$
Fin : $\underline{f} := \min_{x \in Res} \underline{\boldsymbol{f}(\boldsymbol{x})}$, $\overline{f} := \min_{x \in Res} \overline{\boldsymbol{f}(\boldsymbol{x})}$
le min et le max étant pris sur les pavés pour lesquels il est prouvé
qu'ils contiennent un point satisfaisant les contraintes.

3.4 Faiblesse : le résultat dépend de l'expression utilisée

À partir des opérations et fonctions définies sur des intervalles au §2, 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 $f(\boldsymbol{x})\ldots$

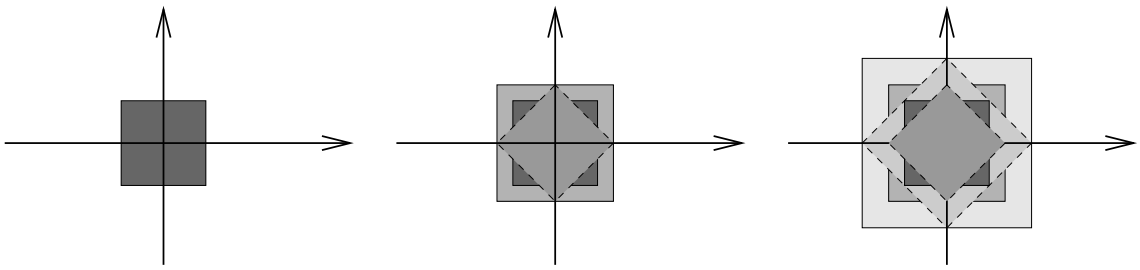

FIG. 4 – Effet enveloppant : après deux rotations successives de $\pi/4$ du petit carré central, on ne retrouve pas le petit carré central mais le grand carré.

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.

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 4 : 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 $f(\boldsymbol{x})$.

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 $\{x * y + x' * z \mid x \in \boldsymbol{x}, x' \in \boldsymbol{x}, y \in \boldsymbol{y}, z \in \boldsymbol{z}\}$ 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 [31].
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 $f(\boldsymbol{x})$.

3.5 Faiblesse : tous les problèmes (ou presque) sont NP-durs

On vient de voir que le résultat d'un calcul dépend de l'expression utilisée. Or on souhaite obtenir un résultat précis, et dans l'idéal le plus petit résultat possible, c'est-à-dire le plus petit intervalle (au sens de l'inclusion) qui contient le résultat exact, ce dernier pouvant être un ensemble de forme quelconque. Cet espoir est une utopie dans la majorité des cas, puisque beaucoup de problèmes sont NP-durs. Nous allons établir une liste d'exemples, pour montrer l'ampleur de la classe des problèmes NP-durs en arithmétique par intervalles.

Un premier théorème, dû à Gaganov [13], établit que le problème de l'évaluation à ε près d'une fonction polynomiale à plusieurs variables et à coefficients rationnels, donc l'un des problèmes apparemment les plus simples que l'on puisse envisager, est NP-dur. Pour une fonction quelconque, l'évaluation optimale de cette fonction sur un intervalle est donc un but inaccessible.

Pour ce qui touche à l'algèbre linéaire, J. Rohn et ses co-auteurs ont établi l'appartenance à la classe des problèmes NP-durs de bon nombre de problèmes usuels. En voici une liste non exhaustive et en vrac :

- déterminer si une matrice intervalle est régulière, c'est-à-dire si toutes les matrices ponctuelles qu'elle contient sont inversibles [45] ;
- déterminer si l'ensemble solution de $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ (au sens défini au §6) est borné [42] ;
- calculer un encadrement optimal de cet ensemble solution [46] ;
- calculer un encadrement à $1/4n^4$ près de l'ensemble solution du système linéaire $n \times n$ $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ [44] ;
- calculer la norme d'une matrice $\boldsymbol{A}$ [43] ;
- déterminer la factorisation LU, à ε près, d'une matrice $\boldsymbol{A}$ par l'algorithme d'élimination de Gauss avec pivot partiel [43] ;
- déterminer si un système linéaire rectangulaire $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ admet une solution strictement positive [45] ;
- déterminer s'il existe un système linéaire ponctuel $Ax = b$ avec $A \in \boldsymbol{A}$ et $b \in \boldsymbol{b}$, rectangulaire, qui admet une solution [43] ;
- déterminer l'encadrement optimal de l'ensemble solution du programme linéaire

$$\begin{cases} \min \boldsymbol{c}^t.\boldsymbol{x} & \\ \text{sous les contraintes} & \boldsymbol{A}.\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq 0 \end{cases}$$

  ici le calcul de la borne inférieure est dans P et celui de la borne supérieure est NP-dur [45].
- ...

On vient de le voir, une très grande partie des problèmes usuellement traités sont NP-durs. Cela justifie le fait que la complexité des problèmes étudiés dans la suite de ce cours ne sera jamais mentionnée, sauf exception.

Comme on peut le constater avec les premiers algorithmes vus ici, d'optimisation globale, et comme cela se confirmera par la suite, le seul argument permettant d'établir facilement la terminaison des algorithmes est le fait que les intervalles traités sont coupés en deux à chaque étape et qu'ils finiront donc par avoir une largeur inférieure à ε, si on prend soin de les couper selon leur plus grand côté. Cet argument conduit à une complexité exponentielle des algorithmes proposés (ce qui rejoint le caractère NP-dur des problèmes). En pratique, on cherche des procédures de réduction les plus efficaces possibles afin de ne couper les pavés en deux qu'en dernier recours, même si les étudier de façon théorique pour affiner les bornes de complexité reste une tâche ardue.

4 Évaluation d'une expression : développements de Taylor

La partie précédente a mis en évidence les avantages uniques de l'arithmétique par intervalles, au travers des algorithmes pour l'optimisation globale d'une fonction continue, mais également ses faiblesses, l'une d'elles étant la dépendance vis-à-vis de l'expression utilisée pour le calcul.

Déterminer l'expression qui conduira au meilleur résultat est un problème difficile et on ne sait pas encore très bien comment l'aborder. En revanche, à partir d'une expression donnée, on peut montrer que pour des intervalles pas trop larges, l'utilisation de développements de Taylor d'ordres 1 et 2 réduit le phénomène de surestimation. Le principe fondamental consiste à utiliser une

expression qui fait apparaître le plus de calculs scalaires possibles et peu de calculs par intervalles. C'est ce qui est présenté au §4.2. Mais commençons par définir l'extension d'une fonction, ou plutôt d'une expression, à des entrées intervalles et les propriétés désirables d'une telle extension. La suite de cette section est reprise intégralement de [39].

4.1 Évaluation des fonctions : fonctions d'inclusion naturelles

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[4] 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 §4.2 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

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

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 :

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

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 $q(\boldsymbol{x}, \boldsymbol{y}) = \max(|\underline{x} - \underline{y}|, |\overline{x} - \overline{y}|)$ 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.

Un premier théorème encourageant a été donné par Moore dans [31], on peut également le trouver dans [33] : sous certaines conditions assez classiques sur le caractère lipschitzien de l'expression à évaluer, l'écart entre l'intervalle calculé et l'image de la fonction correspondante est proportionnel au diamètre de l'intervalle d'entrée.
Théorème. Soit f une fonction en n variables définie par une expression arithmétique, lipschitzienne au sens intervalle en $\boldsymbol{x}_0 \in \mathbb{IR}^n$, on a :

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

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. §5.2 et 3.2 — consiste à découper le pavé $\boldsymbol{x}$ en petits sous-pavés $\boldsymbol{x}_i$. L'image

[4] Cette égalité est vraie tant que le calcul est exact. Une implantation basée sur l'arithmétique flottante ne vérifiera pas cette égalité puisque, à cause de la prise en compte des erreurs d'arrondi, $\boldsymbol{f}(\{x\})$ sera un intervalle non ponctuel dans le cas général.

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 un nombre de sous-pavés exponentiel en le nombre de variables. Cela rejoint le théorème de Gaganov sur la difficulté d'évaluer précisément une fonction, cf. §4.

D'autres solutions reposent sur l'utilisation des formules de Taylor-Lagrange (l'utilisation des fonctions pente ne sera pas abordée dans ce cours) 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.

4.2 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 (également appelée théorème des accroissements finis) 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 [33]. 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 en pratique 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 [5]. 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 un théorème dû à Caprani et Madsen [10]. Ce n'est plus vrai si le point utilisé pour le développement n'est pas le milieu de l'intervalle.

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.

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 [19]. Par bonheur elle était incluse dans notre boîte à outils d'opérations et de fonctions, cf. §2.2, 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 d'ordre élevé, ce qui explique qu'en pratique on n'utilise que des développements de Taylor d'ordres 1 ou 2. Enfin, pour ces mêmes fonctions multivariées, 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.
En revanche, l'obtention des premières dérivées partielles est maintenant bien maîtrisée, grâce aux progrès de la différentiation automatique [14]. Cependant, les logiciels présentés au §2.4 ne les calculent pas tous. Même parmi les logiciels implantant la différentiation automatique, la qualité des valeurs calculées dépend du mécanisme choisi : la différentiation directe (ou *forward*) est la plus simple à mettre en œuvre tandis que la différentiation automatique régressive (ou *backward*) donne de meilleurs résultats. C'est pour cela que la description sommaire des logiciels mentionnait ces informations.

Cette section était consacrée à l'évaluation par intervalles d'une expression et à quelques techniques qui limitent le phénomène de surestimation, au moins pour des intervalles de faible largeur. La suite de ce cours est consacrée au détail des algorithmes évoqués aux §3.2 et 3.3, c'est-à-dire aux briques de base des algorithmes d'optimisation globale. La première de ces briques de base est l'algorithme de Newton par intervalles.

5 Recherche des zéros d'une fonction : Newton par intervalles

L'algorithme de Newton par intervalles permet de localiser les zéros d'une fonction de classe $\mathcal{C}^1$. Rappelons que dans le cas d'une fonction $f : \mathbb{R} \to \mathbb{R}$, l'itération de Newton est définie comme suite : si x^k est l'itéré courant, le nouvel itéré x^{k+1} est donné par

$$x^{k+1} = x^k - f(x^k)/f'(x^k).$$

Cette formule ne peut pas être transposée telle quelle au calcul par intervalles. En effet, si l'on calcule la largeur de x^{k+1}, on obtient

$$\begin{array}{rcl} w(x^{k+1}) & = & w(x^k) + w\left(f(x^k)/f'(x^k)\right) \\ & \geq & w(x^k), \end{array}$$

autrement dit cette itération diverge.

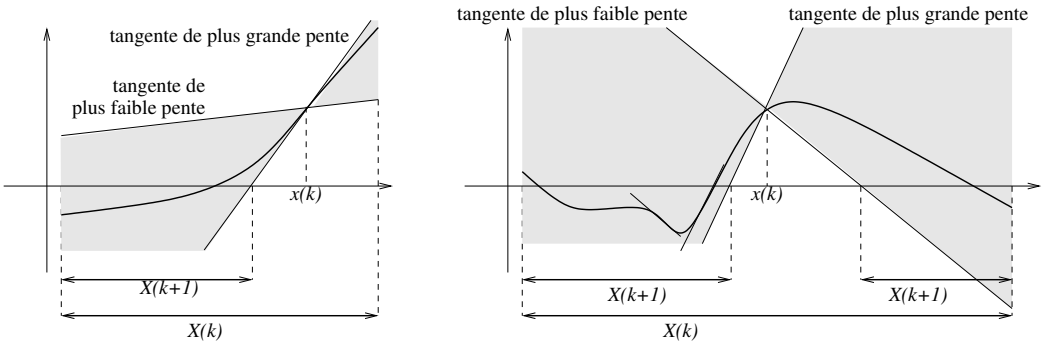

FIG. 5 – Schéma de la méthode de Newton dans le cas d'une seule variable.

5.1 Présentation de la méthode de Newton par intervalles : cas univarié

Pour avoir convergence, il faut donc déterminer une itération contractante. Pour cela, il faut appliquer le principe général déjà énoncé : déterminer une formule comportant un maximum de calculs scalaires et un minimum de calculs sur les intervalles. C'est possible pour l'itération de Newton et cela s'illustre aisément par la figure 5.

On cherche les zéros de la fonction f, c'est-à-dire les points d'intersection de la courbe représentative de f avec l'axe des abscisses. Comme il s'agit de calcul par intervalles, il faut garantir de ne pas oublier un seul de ces zéros. Pour cela, la courbe représentative de f est remplacée par le cône grisé sur la figure. Ce cône est défini par le point[5] $(x, f(x))$ avec x une abscisse quelconque appartenant à l'intervalle considéré $\boldsymbol{x}$ et ses pentes sont données par les extrêma des dérivées de f sur $\boldsymbol{x}$. En effet, la formule de Taylor-Lagrange indique que pour tout $x' \in \boldsymbol{x}$, $\exists \xi \in [x, x']$ et *a fortiori* $\exists \xi \in \boldsymbol{x}$ vérifiant

$$f(x') = f(x) + f'(\xi).(x' - x) \in f(x) + f'(\boldsymbol{x})(\boldsymbol{x} - x).$$

Il suffit donc d'avoir un intervalle contenant $f'(\boldsymbol{x})$, qui peut être obtenu en évaluant une expression pour f' sur $\boldsymbol{x}$, pour pouvoir déterminer le cône grisé de la figure. Cette expression pour f' peut éventuellement être donnée par un mécanisme de différentiation automatique de f.

L'ensemble des zéros de f, c'est-à-dire des points d'intersection de la courbe représentative de f avec l'axe des abscisses, est inclus dans l'intersection de ce cône grisé avec l'axe des abscisses et cette intersection est facile à déterminer. En effet, tout point (x', y') du cône se trouve sur la droite passant par $(x, f(x))$ et par lui-même et l'équation de cette droite est $y' - f(x) = \alpha(x' - x)$ avec $\alpha \in f'(\boldsymbol{x})$. En particulier, tout point du cône qui se trouve sur l'axe des abscisses vérifie à la fois cette équation et $y' = 0$. On obtient donc que l'abscisse x' d'un tel point vérifie

$$-f(x) = \alpha(x' - x) \text{ avec } \alpha \in f'(\boldsymbol{x})$$

ou encore

$$x' = x - \frac{f(x)}{\alpha}.$$

L'ensemble des points d'intersection du cône grisé avec l'axe des abscisses est donc donné par

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

Cette formule ressemble à celle de l'itération de Newton. On note cependant qu'un seul intervalle intervient, comme argument de f'.

[5]Dans le cas d'une implantation avec une arithmétique approchée, le point $(x, f(x))$ devra être remplacé par le segment $(x, \boldsymbol{f}(x))$ puisque, à cause des erreurs d'arrondi, l'évaluation de $f(x)$ produira un intervalle.

L'algorithme de Newton par intervalles dans le cas d'une seule variable est donné ci-dessous.

Données : $\boldsymbol{f}$ une extension intervalle de f et $\boldsymbol{f}'$ une extension intervalle de f'
$\boldsymbol{x}_0$ un pavé de recherche
Résultat : Res une liste de pavés susceptibles de contenir un zéro de f
Initialisations :
$\mathcal{L} := \{\boldsymbol{x}_0\}$ la liste des pavés en attente de traitement
Res $:= \emptyset$
tant que $\mathcal{L} \neq \emptyset$ **faire**
sortir un pavé $\boldsymbol{x}$ de $\mathcal{L}$
choisir x un point quelconque de $\boldsymbol{x}$, par exemple $x = \text{mid}(\boldsymbol{x})$
$N(x, \boldsymbol{x}) = x - \boldsymbol{f}(x)/\boldsymbol{f}'(\boldsymbol{x})$
$\boldsymbol{x}' = N(x, \boldsymbol{x}) \cap \boldsymbol{x}$
si $\boldsymbol{x}' \neq \emptyset$ et $\boldsymbol{f}(\boldsymbol{x}') \ni 0$ alors
ranger $\boldsymbol{x}'$ dans $\mathcal{L}$ ou Res

Dans l'algorithme qui précède, deux points sont à commenter. Tout d'abord, on prend pour nouvel itéré l'intersection de l'intervalle calculé avec le précédent itéré, pour limiter la recherche à l'intervalle $\boldsymbol{x}_0$ initialement considéré. On voit en effet sur la figure que, sans cette intersection, le nouvel itéré pourrait déborder du cadre prescrit. C'est ce qui rend l'itération contractante. Ensuite, on utilise à nouveau une structure de liste pour les intervalles en attente de traitement, même si on part d'un seul intervalle. On voit sur la partie droite de la figure que dans le cas où f' s'annule sur $\boldsymbol{x}$, l'intersection du cône grisé avec l'axe des abscisses est constitué de deux parties disjointes et donc $N(x, \boldsymbol{x})$ est composé de deux intervalles. Dans ce dernier cas, comme $\boldsymbol{f}'(\boldsymbol{x})$ contient 0, la division n'est pas celle définie au §2.2 et il s'agit d'une division dite étendue [22, 38], définie comme suit : si $\boldsymbol{x} = [\underline{x}, \overline{x}]$ et $\boldsymbol{y} = [\underline{y}, \overline{y}]$,

$$\frac{\boldsymbol{x}}{\boldsymbol{y}} = \begin{cases} \left[\overline{x}/\underline{y}, +\infty\right] & \text{si } \overline{x} \leq 0 \text{ et } \overline{y} = 0 \\ [-\infty, \overline{x}/\overline{y}] \cup \left[\overline{x}/\underline{y}, +\infty\right] & \text{si } \overline{x} \leq 0 \text{ et } \underline{y} < 0 < \overline{y} \\ [-\infty, \overline{x}/\overline{y}] & \text{si } \overline{x} \leq 0 \text{ et } \underline{y} = 0 \\ [-\infty, +\infty] & \text{si } \underline{x} < 0 < \overline{x} \\ \left[-\infty, \underline{x}/\underline{y}\right] & \text{si } \underline{x} \geq 0 \text{ et } \overline{y} = 0 \\ \left[-\infty, \underline{x}/\underline{y}\right] \cup [\underline{x}/\overline{y}, +\infty] & \text{si } \underline{x} \geq 0 \text{ et } \underline{y} < 0 < \overline{y} \\ [\underline{x}/\overline{y}, +\infty] & \text{si } \underline{x} \geq 0 \text{ et } \underline{y} = 0 \end{cases}$$

En pratique, quand on implante l'algorithme de Newton, il reste à préciser le test d'arrêt et également, pour prouver la terminaison de l'algorithme, à garantir que l'itération est suffisamment contractante. Un test d'arrêt couramment utilisé consiste à exiger que la largeur de $\boldsymbol{x}$ soit inférieure à un seuil ε_x, autrement dit que $\boldsymbol{x}$ soit un encadrement précis d'un zéro de f, ou alors que la largeur de $\boldsymbol{f}(\boldsymbol{x})$ soit suffisamment faible, $|\boldsymbol{f}(\boldsymbol{x})| \leq \varepsilon_y$: dans ce cas, f est considérée comme suffisamment proche de 0. Pour garantir que l'itération est suffisamment contractante, on coupe éventuellement l'intervalle $\boldsymbol{x}$ en deux si le nouvel itéré $\boldsymbol{x}'$ n'est pas considéré comme suffisamment réduit.

5.2 Présentation de la méthode de Newton par intervalles : cas multivarié

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 :

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

Plus précisément, on détermine une solution $N(\tilde{x}, \boldsymbol{x}^k)$ du système linéaire $\boldsymbol{f}'(\boldsymbol{x}^k)(\tilde{x} - N(\tilde{x}, \boldsymbol{x}^k)) = f(\tilde{x})$ et le nouvel itéré est $\boldsymbol{x}^{k+1} = N(\tilde{x}, \boldsymbol{x}^k) \cap \boldsymbol{x}^k$. Ici encore, l'itération ainsi construite est clairement monotone décroissante pour l'inclusion puisque par construction $\boldsymbol{x}^{k+1} \subset \boldsymbol{x}^k$.

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. Une méthode couramment utilisée est la méthode de Gauss-Seidel avec pré-multiplication par une matrice C (cf. §6.3). On peut montrer que si $C\boldsymbol{A}$ est une H-matrice et si $\tilde{x} = \text{mid}(\boldsymbol{x})$ alors cette itération 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*.

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, il est plus détaillé que l'algorithme qui était présenté de façon sommaire au §5.1 pour éviter les redites.

Algorithme de Newton par intervalles

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 $\mathcal{L}$
$\boldsymbol{x}' := N(\tilde{x}, \boldsymbol{x})$ avec N un pas d'une méthode de Newton
si $\boldsymbol{x}' \neq \emptyset$ alors
si $N(\tilde{x}', \boldsymbol{x}') \subset \boldsymbol{x}'$ alors ¡¡ $\boldsymbol{x}'$ contient un zéro ¿¿ // cf. §5.3
si $N(\tilde{x}', \boldsymbol{x}') \subset \text{int}(\boldsymbol{x}')$ alors ¡¡ $\boldsymbol{x}'$ contient un unique zéro ¿¿
si $\boldsymbol{x}'$ satisfait le critère d'arrêt alors
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 [4]. Le test sur la taille de $\boldsymbol{x}'$ mesure la précision relative obtenue pour le zéro contenu dans $\boldsymbol{x}'$, il 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 [4, 40], 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 une valeur de 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 [17, 37]. 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.

5.3 Propriétés de la méthode de Newton par intervalles : preuve de l'existence et de l'unicité d'un zéro

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 §4.1), 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/Schauder. Il est remarquable que les hypothèses de ce théorème soient faciles à vérifier en arithmétique par intervalles : comparer $\boldsymbol{x}$ et un surencadrement de $N(\tilde{x}, \boldsymbol{x})$ 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 $N(\tilde{x}, \boldsymbol{x})$ 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 [47] 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 $N(\tilde{x}, \boldsymbol{x}_\varepsilon)$ 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 ¡¡

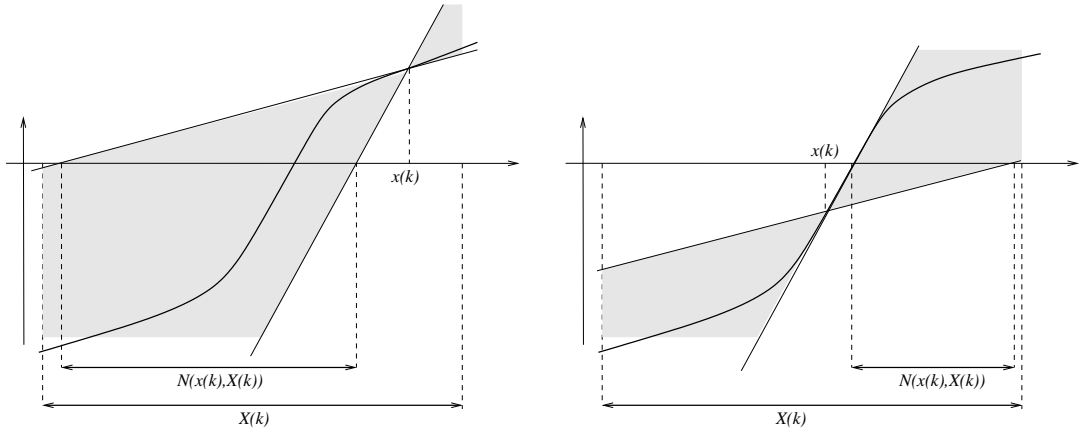

FIG. 6 – Influence du ¡¡ gonflage ¿¿ sur l'applicabilité du théorème de Moore et Nickel : à gauche, le point $\tilde{x}$ est au bord de l'intervalle et le meilleur $N(\tilde{x}, \boldsymbol{x})$ est presque égal à $\boldsymbol{x}$, à droite le point $\tilde{x}$ est plus loin des bords après ¡¡ gonflage ¿¿ de $\boldsymbol{x}$ et le meilleur $N(\tilde{x}, \boldsymbol{x})$ est plus réduit, ce qui correspond à de meilleures chances que $N(\tilde{x}, \boldsymbol{x})$ soit inclus dans l'intérieur de $\boldsymbol{x}_\varepsilon$ en pratique.

gonflage ¿¿ soit plus net, cf. [28]. Cela permet de recentrer $\tilde{x}$ et de réduire $N(\tilde{x}, \boldsymbol{x})$, comme indiqué sur la figure 6.

5.4 Cas de plusieurs zéros proches ou d'un zéro multiple

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 et u la précision machine : 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. [2] sur l'importance d'une bonne évaluation de la fonction dans la méthode de Newton par intervalles) permet de distinguer ces racines.

6 Résolution d'un système linéaire par intervalles

Cette partie doit beaucoup à l'ouvrage de Neumaier [33] pour les algorithmes et leur analyse. Il ne sera donc pas cité systématiquement par la suite.

6.1 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 $\boldsymbol{b} \in \mathbb{IR}^n$, il n'existe pas nécessairement de vecteur $\boldsymbol{x} \in \mathbb{IR}^n$ vérifiant[6] $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.

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

[6] 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 $\boldsymbol{b}$.

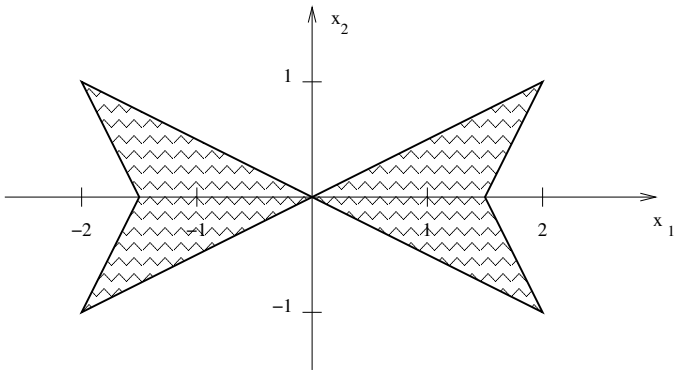

FIG. 7 – Solution d'un système linéaire intervalle.

n'est pas nécessairement convexe, comme le montre l'exemple suivant [33] : 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 7.

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

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

Pour mémoire, il a été mentionné au §3.5 que le problème de déterminer exactement $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b})$ est NP-dur et le déterminer à un polynôme en n près l'est également.

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 $\Sigma_{\exists\exists}(\boldsymbol{A}, \boldsymbol{b})$.

6.2 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 $\boldsymbol{LU} = \{L \times U \mid L \in \boldsymbol{L}, U \in \boldsymbol{U}\}$ 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 $\boldsymbol{Ux} = \boldsymbol{y}$.

Appliquer une stratégie de pivot partiel ne permet pas nécessairement de stabiliser l'algorithme de Gauss, comme l'illustre l'exemple suivant :

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

$\boldsymbol{A}$ 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 (voir [33] pour les définitions et les preuves).

L'algorithme d'élimination de Gauss par intervalles peut conduire à des surencadrements du résultat beaucoup trop larges, en particulier quand les éléments de la matrice sont des intervalles larges. On a même vu au §3.5 que la surestimation peut être arbitrairement grande. Illustrons ce phénomène par un exemple :

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

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

Cet algorithme est revenu en usage après qu'un pré-traitement[7] lui a été adjoint. L'idée d'un pré-traitement consiste à rapprocher la matrice du système linéaire à résoudre de la matrice identité. On peut songer *a priori* à pré-multiplier 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 $\boldsymbol{x} = C'\boldsymbol{z}$; le choix $C = \text{mid}(\boldsymbol{A})^{-1}$, $C' = I$ est le pré-traitement le plus utilisé en pratique[8].

Dans le cas où $\text{mid}(\boldsymbol{A})^{-1} \times \boldsymbol{A}$ est régulière, l'algorithme d'élimination de Gauss peut lui être appliqué. Si de plus $\text{mid}(\boldsymbol{A})^{-1} \times \boldsymbol{A}$ 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 [29] ont montré que dans le cas où la matrice milieu de cette matrice est diagonale, alors un autre algorithme direct, l'algorithme de Hansen-Bliek-Rohn-Ning-Kearfott [34], 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 pré-traités puis qu'il soit prouvé optimal pour les M-matrices.

6.3 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 :

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

[7] Ce pré-traitement est souvent et improprement appelé ¡¡ préconditionnement ¿¿.

[8] Comme en général les problèmes abordés en arithmétique par intervalles sont de petite taille, comparée à celle des problèmes résolus en utilisant l'arithmétique flottante usuelle, l'inversion d'une matrice ponctuelle par une procédure numérique n'est pas déraisonnable.

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

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

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 :

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

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), 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.

Ici aussi, un pré-traitement peut permettre d'améliorer le comportement de l'algorithme, le plus souvent il consiste à multiplier le système à gauche par $\mathrm{mid}(\boldsymbol{A})^{-1}$. L'algorithme itératif de Gauss-Seidel ainsi pré-traité porte le nom d'algorithme de Rump [48] ou de Hansen-Sengupta. Pour une étude des pré-multiplieurs pour l'algorithme de Gauss-Seidel, voir [3].

Dans le cas où la matrice du système éventuellement pré-traité $\boldsymbol{A}$ 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 alors 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 [50] 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.

7 Résolution de contraintes

La résolution de systèmes linéaires et non linéaires a également été étudiée par des spécialistes de la programmation logique sous contraintes dans le cas discret. Ils ont adapté les techniques et les notions utilisées en programmation logique au cas du calcul sur les intervalles [54, 8, 11, 21].

7.1 Algorithme de propagation - rétropropagation

Une technique couramment utilisée en programmation logique pour résoudre un ensemble de contraintesest 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 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 [55], Prolog IV [7] et Alias [12].

Cet algorithme est un exemple de contracteur [21] : ce sont les algorithmes employés en programmation logique sous contraintes. L'algorithme de Gauss-Seidel présenté au §6.3 peut aussi être vu comme un contracteur. Cet algorithme n'utilise que la phase de rétropropagation et ne met à jour que la variable $\boldsymbol{x}_i$ à l'aide de la i-ème contrainte. Les contracteurs ou algorithmes contractants réduisent l'ensemble initial sans jamais procéder à des bissections et ils ont des complexités polynomiales. Cependant, ils s'arrêtent quand ils ne parviennent plus à réduire l'ensemble courant et non quand ils sont parvenus à l'optimum. Un bon contracteur est donc un algorithme qui s'arrête avec un ensemble petit au sens de l'inclusion. Les notions présentées au §7.2 permettent de comparer les ensembles obtenus et par conséquent les contracteurs.

Si jamais on désire réduire encore le résultat obtenu par un contracteur, on peut sacrifier la complexité polynomiale et procéder à une bissection. On applique ensuite le contracteur aux deux sous-pavés pour les réduire à nouveau etc.

La méthode de Newton tout comme la méthode de propagation des contraintes, et les contracteurs de façon générale, sont, comme leur nom l'indique, 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 et ces algorithmes, puisqu'il est facile de vérifier une inclusion telle que $\boldsymbol{f}(\boldsymbol{x}) \subset \boldsymbol{x}$ ou $\boldsymbol{f}(\boldsymbol{x}) \subset \text{int}(\boldsymbol{x})$ pour pouvoir en conclure à l'existence et éventuellement à l'unicité du résultat cherché.

7.2 Notions adaptées de la programmation logique sous contraintes

Outre les techniques utilisées en programmation logique sous contraintes, quelques notions ont été importées. En particulier les notions de consistance ont été adaptées. Elles permettent de classifier les méthodes employées et de les comparer, ou, comme montré dans l'exemple ci-dessous, de comparer la qualité des intervalles résultats. [54, 8, 11].

Par exemple, une contrainte de la forme $c(x_1, \ldots, x_n)$ est dite consistante par arc avec l'ensemble E, qui est supposé être le résultat calculé, 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 x_j avec $j \neq i$ 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*.

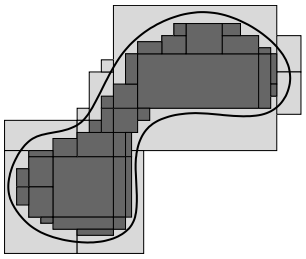

FIG. 8 – Illustration des encadrements intérieur et extérieur.

Une contrainte de la forme $c(x_1, \ldots, x_n) = 0$ avec $x_i \in \mathbb{R}$ pour $1 \leq i \leq n$ est dite consistante par pavé avec le pavé $\boldsymbol{B} = \boldsymbol{B}_1 \times \ldots \boldsymbol{B}_n \in \mathbb{IR}^n$, $\boldsymbol{B}_i \in \mathbb{IR}$ 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 l'enveloppe convexe fermée de l'ensemble pour lequel la contrainte est consistante par arc.

8 Conclusion

8.1 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 [6] 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 SIVIA proposée par Jaulin [21] et illustrée figure 8 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. Le problème de Cauchy est la recherche d'une solution x satisfaisant :

$$\left\{ \begin{array}{lcl} x' & = & f(t, x), \\ x(t_0) & = & x_0. \end{array} \right.$$

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 [32].

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.

8.2 Conclusion et credo personnel

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, le plus souvent des algorithmes itératifs basés sur une itération contractante, 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. En effet, dès lors que l'on dispose d'une itération contractante, seule la précision du calcul (en faisant fi de la complexité exponentielle du calcul) constitue un obstacle à l'obtention de résultats aussi précis que l'on veut. 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.* [21] en fait foi, tout comme les travaux du groupe de travail *Méthodes ensemblistes pour l'automatique*, cf. `http://www-lag.ensieg.inpg.fr/gt-ensembliste/`.

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

[2] 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.

[3] 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.

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

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

[6] 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.

[7] 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.

[8] 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.

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

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

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

[12] Inria Coprin project. ALIAS (Algorithms Library of Interval Analysis for Systems). `http://www-sop.inria.fr/coprin/developpements/main.html`, 2003.

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

[14] A. Griewank. *Evaluating Derivatives - Principles and Techniques of Algorithmic Differentiation*. SIAM, 2000.

[15] M. Grimmer, K. Petras, and N. Revol. Multiple precision interval packages : Comparing different approaches. In *Lecture Notes in Computer Science*, volume 2991, pages 64–90, 2004.

[16] G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann. The MPFR library (v. 2.0.3). `http://www.mpfr.org`, 2004.

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

[18] B. Hayes. A Lucid Interval. *Scientific American*, 91(6) :484–488, 2003.

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

[20] IBM. *IBM High-Accuracy Arithmetic Subroutine Library (ACRITH)*, 1986.

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

[22] W. Kahan. A more complete interval arithmetic. Lecture notes for a summer course at the University of Michigan, 1968.

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

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

[25] O. Knueppel. PROFIL - Programmer's Runtime Optimized Fast Interval Library. Technical Report Bericht 93.4, Technische Universität Hamburg-Harburg, 1993.

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

[27] M. Lerch, G. Tischler, J. Wolff von Gudenberg, W. Hofschuster, and W. Krämer. *The interval library filib++ 2.0*. Universität Wuppertal, Germany, 2001.

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

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

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

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

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

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

[34] 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.

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

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

[37] 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.

[38] D. Ratz. On extended interval arithmetic and inclusion isotonicity. Technical report, Institut für Angewandte Mathematik, Universität Karlsruhe, Germany, 1996.

[39] N. Revol. Arithmétique par intervalles. *Calculateurs Parallèles*, 13 :387–426, 2001.

[40] N. Revol. Interval newton iteration in multiple precision for the univariate case. *Numerical Algorithms*, 34(2) :417–426, 2003.

[41] N. Revol and F. Rouillier. The MPFI library (v. 1.3.2). `http://perso.ens-lyon.fr/nathalie.revol/software.html`, 2004.

[42] J. Rohn. Interval matrices : singularity and real eigenvalues. *SIAM J. Matrix Anal. Appl.*, 14(1) :82–91, January 1993.

[43] J. Rohn. Complexity of solving linear interval equations. Technical Report 636, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, May 1995.

[44] 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.

[45] J. Rohn. Complexity of some linear problems with interval data. Technical Report 687, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, October 1996.

[46] 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.

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

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

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

[50] Y. Saad. *Iterative methods for sparse linear systems*. PWS Publishing, 1996.

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

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

[53] T. Sunaga. Theory of interval algebra and its application to numerical analysis. *RAAG Memoirs, Ggujutsu Bunken Fukuy-kai. Tokyo*, 2 :29–46, 1958.

[54] 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.

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

[56] R. C. Young. The algebra of many-valued quantities. *Mathematische Annalen*, 104 :260–290, 1931.