Université de Perpignan Via Domitia
Laboratoire de Physique Appliquée et d'Automatique
Équipe de recherche en Informatique DALI

THÈSE

pour obtenir le grade de

Docteur de l'université de Perpignan
spécialité : Informatique

au titre de l'école doctorale BESPI (ED 305)

présentée et soutenue publiquement le 30 novembre 2005

par Stef GRAILLAT

Fiabilité des algorithmes numériques : pseudosolutions structurées et précision

Devant la commission d'examen formée de

Marc	DAUMAS	Chargé de recherche CNRS, HDR	LIRMM	Examinateur
Philippe	LANGLOIS	Professeur	U. Perpignan	Directeur de thèse
Siegfried	RUMP	Prof. Dr. Habil.	TU Hambourg	Rapporteur
Gilles	VILLARD	Chargé de recherche CNRS, HDR	LIP	Rapporteur
Éric	WALTER	Directeur de recherche CNRS	L2S	Président du jury
Paul	ZIMMERMANN	Directeur de recherche INRIA	LORIA	Examinateur

À ma famille, à mes amis

Remerciements

En premier lieu, je tiens à remercier mon directeur de thèse Philippe Langlois sans qui cette thèse n'aurait jamais pu voir le jour. Je le remercie pour son encadrement, ses conseils et pour sa générosité aussi bien dans le travail que dans la vie de tous les jours.

Je tiens à remercier Gilles Villard et Siegfried M. Rump d'avoir accepté la lourde tâche de rapporter cette thèse malgré les délais très courts que je leur ai imposés et de m'avoir permis, par leurs commentaires avisés, d'améliorer la qualité de ce document.

Je remercie Marc Daumas, Éric Walter et Paul Zimmermann de m'avoir fait l'honneur d'être membre de mon jury. J'apprécie la qualité des nombreuses remarques qu'ils ont pu me faire sur ce travail.

J'ai une pensée toute particulière pour tous les membres de l'équipe DALI, Vincent Beaudenon, Marc Daumas, David Defour, Bernard Goossens, Philippe Langlois, Nicolas Louvet, David Parello, pour leur aide constante durant cette thèse.

Je remercie mes cobureaux successifs que ce soit lors de mon DEA au LIP, Sylvie Boldo et Nicolas Boullis et à Perpignan, Éric Dubon, Viorica Venera Motreanu, David Defour, Benoît Bréholée, Nicolas Louvet et Vincent Beaudenon.

Je remercie aussi tous les membres de l'équipe Arénaire qui m'ont toujours très bien accueilli lors de mes différentes visites. Merci en particulier à Nicolas Brisebarre, Florent de Dinechin, Claude-Pierre Jeannerod, Jean-Michel Muller, Nathalie Revol, Arnaud Tisserand, Gilles Villard.

Je remercie aussi tous mes camarades de l'ENS de Cachan, antenne de Bretagne avec qui j'ai passé de très bons moments. Merci à vous Florence Bachmann, Vincent Beck, Sylvain Brochard, Matthieu Gendulphe, Jérôme Malick, Gabriel Peyré, Karel Pravda-Starov, Jean Starynkevitch.

Je remercie particulièrement Françoise Tisseur pour m'avoir proposé de travailler avec elle et pour m'avoir ensuite invité quelques mois à Manchester après la thèse. Un merci

spécial à Nicolas Louvet avec qui s'est un véritable plaisir de travailler.

Enfin, je remercie ma famille pour leur compréhension et leur soutien sans faille tout au long de ces trois années.

Table des matières

Liste des figures

Liste des tableaux

Liste des algorithmes

Notations

$:=$	définition.
$\mathbf{C}$	le corps des nombres complexes.
$\mathbf{R}$	le corps des nombres réels.
$\mathbf{K}$	un corps (habituellement $\mathbf{R}$ ou $\mathbf{C}$).
$\mathbf{K}[x]$	l'anneau de polynômes d'indeterminé x à coefficients dans $\mathbf{K}$.
$M_n(\mathbf{K}), \mathbf{K}^{n\times n}$	matrice de taille (n, n) sur le corps $\mathbf{K}$.
$\mathcal{P}_n(\mathbf{K})$	polynôme à coefficients dans $\mathbf{K}$ de degré au plus n.
$\overline{z}$	le conjugé du nombre complexe z.
$\underline{z}$	le vecteur $(1, z, z^2, \ldots, z^n)$.
x^T	la transposé du vecteur x.
x^*	le transconjugé du vecteur x.
$\|\cdot\|_*$	la norme duale de $\|\cdot\|$: $\|y\|_* = \sup_{\|x\|=1} \|y^*x\|$.
$\|\cdot\|_p$	la norme p de Hölder : $\|x\|_p = (\sum_{i=1}^n \|x_i\|^p)^{1/p},\ 1 \leqslant p \leqslant \infty$.
$\mathbf{u}$	unité d'arrondi.
$\underline{\mathbf{u}}$	unité d'underflow.

Introduction

Comment valider le résultat d'un algorithme numérique de résolution d'un problème mathématique ? Comment apprécier la difficulté de résoudre un problème ? Comment mesurer la qualité d'un algorithme de résolution d'un problème donné ? Telles sont les questions auxquelles nous nous intéressons dans cette thèse.

1 Motivations

Calculer la solution de la modélisation mathématique d'un problème donné nécessite d'étudier les effets de quatre types d'erreurs.

- Les *erreurs de modèle* mesurent la différence entre le problème physique et la formulation mathématique. Grâce à l'augmentation importante de la puissance des calculateurs, on essaie aujourd'hui de prendre des modèles de plus en plus proches de la réalité physique et donc de plus en plus complexes. Cela permet de diminuer ces erreurs de modèle.
- Les *erreurs de troncature* sont introduites par le schéma numérique. Beaucoup de méthodes numériques (comme par exemple la méthode des trapèzes pour le calcul d'intégrales, la méthode d'Euler pour les équations différentielles ou bien la méthode de Newton pour les équations non-linéaires) correspondent à ne sommer qu'un nombre fini de termes d'une série numérique. Les termes omis constituent l'erreur de troncature. L'analyse des erreurs de troncature est la tâche principale de l'analyse numérique.
- Les *erreurs de donnée* apparaissent lorsque l'on résout un problème pratique. Elles proviennent alors souvent d'erreurs de mesure ou d'estimations (qui peuvent être importantes en ingénierie et en économie, par exemple de l'ordre de quelques chiffres). Elles peuvent aussi provenir d'erreurs dues au stockage de ces valeurs dans un ordinateur. Enfin, elles peuvent aussi provenir d'erreurs dans les calculs précédents si les données sont elles-mêmes les solutions d'un autre problème.
- Les *erreurs d'arrondi* sont une conséquence du travail en précision finie. On approche l'ensemble des nombres réels par un ensemble fini, par exemple l'ensemble des nombres flottants. Prendre en compte les différences entre l'arithmétique réelle

et l'arithmétique en précision finie est une préoccupation importante de la recherche en arithmétique des ordinateurs.

Ces différentes erreurs conditionnent la précision de la solution calculée. Nous ne traiterons pas ici des erreurs de modèle. Celles-ci tiennent plus du domaine du problème considéré et dépendent fortement de ce problème. Nous n'étudierons pas non plus les erreurs de troncature qui relèvent plutôt de l'analyse numérique. Dans cette thèse, nous nous intéressons donc plus particulièrement aux effets des *erreurs de données et d'arrondi*.

Les outils classiques de l'analyse d'erreur en précision finie sont le *conditionnement*, l'*erreur inverse* et l'*erreur directe*. Le conditionnement mesure la difficulté de résoudre un problème et ce indépendamment de la méthode utilisée pour résoudre ce même problème. L'erreur inverse mesure la distance entre le problème que l'on a effectivement résolu (étant entendu qu'avec les erreurs d'arrondi nous n'avons pas obtenu le résultat exact) et le problème initial. L'erreur directe mesure elle l'écart entre la solution exacte de notre problème et la solution effectivement calculée. Pour les cas où l'erreur directe est difficile à calculer voire à évaluer, on étudie le conditionnement et l'erreur inverse. Cela est motivé par l'estimation empirique de l'erreur directe suivante,

$$\text{erreur directe} \lesssim \text{conditionnement} \times \text{erreur inverse},$$

autrement appelée la « rule of thumb ».

Nous utilisons aussi la notion de *pseudosolutions* : dans cette thèse, il s'agit des *pseudozéros* et du *pseudospectre*. Étant donné un problème, on recherche les solutions des problèmes voisins, c'est-à-dire, des problèmes perturbés. Contrairement au conditionnement où les perturbations sont infinitésimales, les pseudosolutions sont les solutions du problème perturbé de manière finie. Les pseudosolutions sont donc un outil mieux adapté pour traiter des perturbations plus importantes que celles accessibles par le conditionnement. Cela permet de prendre en compte des incertitudes sur les données qui peuvent être largement plus grandes que celles dues à la précision finie.

On transforme en général des problèmes complexes en des problèmes plus simples que l'on sait résoudre. Ces problèmes sont, plus particulièrement :

- évaluer un polynôme ;
- calculer les racines d'un polynôme ;
- calculer les valeurs propres d'une matrice ;
- résoudre un système d'équations linéaires.

Nous nous intéressons, dans cette thèse, à la difficulté de résoudre ces problèmes ainsi qu'à la qualité des logiciels scientifiques utilisés pour les résoudre.

Les deux premiers sont des problèmes d'*algèbre polynomiale numérique*, sujet auquel Stetter a consacré le livre « Numerical Polynomial Algebra » [186]. Les polynômes jouent un rôle de plus en plus important dans le calcul scientifique modélisant des phénomènes physiques en ingénierie et en sciences naturelles, en particulier en biologie et en médecine. Dans ces domaines, les polynômes fournissent un cadre naturel pour la modélisation de

phénomènes du monde réel qui ne peuvent être décrits par un modèle linéaire de façon adéquate.

Le récent « Computer Algebra Handbook » [68] contient presque 100 pages donnant un état de l'art sur les applications du calcul formel. La plupart de ces applications emploie des polynômes ou des systèmes polynomiaux, souvent avec un grand nombre de variables et fournit des résultats numériques. Les relations entre les applications industrielles et la nécessité de résoudre des systèmes polynomiaux ont été étudiées par le projet européen FRISCO[1] (a Framework for Integrated Symbolic/Numeric Computation, 1996–1999). On peut y lire que les phénomènes de non-linéarité apparaissent soit parce que le modèle linéaire n'est pas réaliste soit parce qu'il n'est pas assez précis. Il est dit aussi dans ce rapport que « les coefficients des polynômes sont généralement réels et que dans la plupart des cas ils proviennent de données expérimentales et donc ne sont connus qu'avec une précision limitée ». Traditionnellement, l'analyse de sensibilité sur les polynômes considère des coefficients complexes et des perturbations complexes de ces mêmes coefficients. L'une des contributions de la thèse est de conserver la structure réelle lorsque les polynômes sont à coefficients réels. Autrement dit, on n'autorise alors que des perturbations réelles des coefficients.

Les deux autres problèmes sont des problèmes d'*algèbre linéaire numérique* qui ont déjà été largement étudiés — voir Higham [87] pour une synthèse. Une de nos contributions repose sur la notion de *problèmes structurés*, notion qui connaît depuis quelques années un engouement croissant [17, 50, 77]. L'idée est d'utiliser la structure du problème afin d'améliorer les algorithmes en utilisant les informations supplémentaires liées à cette structure, ou bien, en implantant des transformations successives qui en préservent la structure. Cela permet par exemple de gagner en temps de calcul et en espace mémoire. La prise en compte de la structure permet-elle systématiquement d'améliorer la résolution d'un problème ? Le lien naturel avec les problèmes précédents de l'algèbre polynomiale repose sur la structure de matrice compagnon. En effet, les zéros d'un polynôme sont aussi les valeurs propres de la matrice compagnon associée. Hinrichsen et Kelb [89] ont présenté une façon de perturber la matrice compagnon tout en gardant cette structure compagnon. On retrouve alors les résultats sur les polynômes d'un point de vue matriciel en structurant le problème. Nous nous sommes alors naturellement intéressés à d'autres structures que la structure compagnon.

2 Contributions

Nous pouvons classer nos contributions en quatre catégories :

1. *Amélioration de la précision*
   Un des enjeux actuels en arithmétique des ordinateurs est d'augmenter la précision des algorithmes tout en travaillant à précision constante donnée. Dans cette thèse, nous proposons de modifier l'algorithme de Horner d'évaluation d'un polynôme afin

[1] On peut trouver le rapport du projet à l'adresse suivante http://www.nag.co.uk/local/projects/FRISCO.html.

que le résultat calculé ait la même précision que s'il avait été calculé par le schéma de Horner classique mais avec une précision interne doublée (cette notion sera précisée dans le chapitre 3, page 37).

En utilisant les mêmes techniques, nous proposons un algorithme de calcul de déterminants 2×2 et 3×3 produisant un résultat avec une précision relative donnée tout en travaillant à précision constante. Cela nous permet de calculer de manière certifiée les prédicats géométriques classiques.

2. *Applications des pseudozéros*
   La notion de pseudozéro a été introduite par Mosier en 1986 pour étudier la sensibilité des racines par rapport aux perturbations sur les coefficients. Il s'agissait alors de modéliser les erreurs d'arrondi qui pouvaient apparaître lors de calculs en précision finie. Pour des perturbations de taille plus importante, les pseudozéros permettent de modéliser des incertitudes sur les coefficients des polynômes provenant par exemple de mesures physiques. Nous montrons dans cette thèse que les pseudozéros peuvent avoir des applications dans d'autres domaines que l'analyse numérique. Nous mettons en évidence par exemple que les pseudozéros permettent de tester la primalité approchée de deux polynômes connus avec une certaine incertitude sur leurs coefficients.

   Nous montrons aussi que les pseudozéros permettent de calculer le rayon de stabilité et la pseudoabscisse d'un polynôme. Ces deux quantités ont des applications importantes en théorie du contrôle.

3. *Prise en compte des perturbations réelles*
   Lorsque l'on travaille en arithmétique flottante IEEE 754 avec des données réelles, on sait pertinemment que les erreurs d'arrondi seront encore des données flottantes donc réelles. Or il se trouve que dans l'analyse de sensibilité, on considère toujours que les perturbations peuvent aussi être des nombres complexes. Dans cette thèse, nous étudions, pour les problèmes de l'évaluation polynomiale et la recherche de racines, des conditionnements et des erreurs inverses réelles, c'est-à-dire n'autorisant que des perturbations réelles des coefficients réels des polynômes concernés. Nous montrons par exemple pour ces problèmes, que le conditionnement réel se situe dans un rapport d'au plus $\sqrt{2}$ avec le conditionnement complexe. Néanmoins, nous montrons que l'erreur inverse réelle peut être significativement plus grande que l'erreur inverse complexe.

4. *Perturbations matricielles structurées*
   Il semble raisonnable pour une matrice structurée, par exemple symétrique, Toeplitz, Hankel, etc., de ne considérer que des perturbations structurées de cette même matrice. Cela se justifie par exemple, par le fait que l'on ne stocke de manière pragmatique que la moitié d'une matrice symétrique, l'autre étant identique. Nous étudions donc la notion de pseudospectre structuré pour les matrices Toeplitz, circulantes et Hankel. Nous montrons qu'il n'y a pas de différence entre le pseudospectre structuré et le pseudospectre classique.

   Nous étudions aussi des conditionnements pour les systèmes linéaires et l'inversion matricielle pour des structures dérivant d'algèbres de Lie et de Jordan. Nous étu-

dions pour quelles sous-classes de ces structures il n'y a pas ou peu de différences entre les conditionnements structurés et non structurés.

3 Présentation détaillée de la thèse

Le chapitre 1 est une introduction à l'analyse d'erreur en précision finie. On présente les outils qui seront utilisés dans la suite de la thèse, en particulier les notions de conditionnement, d'erreur inverse, d'erreur directe et la norme IEEE 754 de l'arithmétique flottante. La thèse est ensuite décomposée en trois parties.

3.1 Partie I : Évaluation polynomiale

L'évaluation de polynômes est nécessaire dans beaucoup d'algorithmes numériques. On peut citer par exemple toutes les méthodes de type Newton pour la recherche de racines de polynômes. Dans cette thèse, nous nous sommes intéressés à trois problèmes liés à l'évaluation polynomiale :

- le conditionnement de l'évaluation polynomiale ;
- l'évaluation précise de polynômes par la méthode de Horner ;
- l'application de la sommation précise en géométrie algorithmique.

3.1.1 Conditionnement structuré de l'évaluation polynomiale (chapitre 2)

Motivation : Le conditionnement de l'évaluation polynomiale consiste à mesurer la variation de l'évaluation si l'on perturbe de manière infinitésimale les coefficients du polynôme. Ce conditionnement est depuis longtemps connu pour des polynômes à coefficients complexes et où l'on autorise des perturbations complexes de ces coefficients (voir par exemple [36, 206]). Il en est de même pour l'erreur inverse.

Contribution : Dans la pratique, les polynômes considérés sont souvent à coefficients réels et les perturbations que peuvent subir ces coefficients sont en général elles aussi réelles. Nous définissons un nouveau conditionnement qui permet la seule prise en compte des perturbations réelles. Nous donnons une formule calculable de ce *conditionnement réel*. Nous montrons surtout que le rapport entre ce conditionnement réel et le conditionnement classique se trouve dans l'intervalle $[1, \sqrt{2}]$. Par conséquent, le conditionnement classique est suffisant pour rendre compte de la difficulté de l'évaluation. Nous définissons en outre une erreur inverse réelle. Nous donnons une formule calculable pour cette erreur inverse réelle. Nous montrons aussi que l'erreur inverse réelle peut être très différente de l'erreur inverse classique.

3.1.2 Évaluation précise de polynômes par la méthode de Horner (chapitre 3)

Motivation : L'algorithme le plus efficace d'un point de vue complexité pour l'évaluation d'un polynôme est l'algorithme de Horner [116, p.519, ex 38]. On sait de plus qu'il est inverse-stable [87]. Néanmoins, l'évaluation polynomiale peut être extrêmement mal

conditionnée ; c'est le cas par exemple lorsque l'on se situe près d'une racine multiple. Or évaluer un polynôme de manière précise peut être une tâche cruciale par exemple pour le critère d'arrêt des méthodes itératives de type Newton pour la recherche de zéros. Les solutions existantes utilisent toutes des bibliothèques multiprécisions qui permettent d'augmenter la précision des calculs mais souvent au prix d'un surcoût très élevé.

Contribution : Nous proposons un algorithme de Horner compensé qui permet d'évaluer de façon précise et rapide un polynôme à coefficients flottants. La précision du résultat calculé est similaire à celle donnée en effectuant l'évaluation classique par le schéma de Horner avec une précision double de la précision courante. Notre algorithme est deux fois plus rapide que les implémentations existantes donnant la même précision pour le résultat. Nous présentons aussi un algorithme pour calculer en arithmétique flottante IEEE 754 une borne d'erreur certifiée. Par certifiée, on entend que la borne calculée majore effectivement l'erreur de l'évaluation par l'algorithme de Horner compensé. Des expérimentations avec des polynômes très mal conditionnés illustrent ces résultats théoriques. Nos algorithmes n'utilisent qu'une seule précision courante et sont portables si l'on suppose que l'arithmétique flottante est conforme au standard IEEE 754.

Ce travail en collaboration avec Philippe Langlois et Nicolas Louvet fait l'objet du rapport de recherche [75]. Il a été soumis à SIAM Journal of Scientific Computing.

3.1.3 Applications de la sommation précise en géométrie algorithmique (chapitre 4)

Motivation : Certains algorithmes en géométrie algorithmique se basent sur des prédicats géométriques comme par exemple déterminer de quel côté d'une ligne se situe un point. C'est par exemple le cas pour l'algorithme de triangulation de Delaunay ou bien la génération de maillage pour les EDP. Si ces prédicats géométriques donnent une mauvaise réponse, le résultat de l'algorithme tout entier peut être très éloigné du résultat attendu. Il est donc crucial de pouvoir tester ces prédicats géométriques de manière robuste. La majeure partie de ces tests se ramène au calcul du signe d'un déterminant de petite taille (disons 3×3 ou 4×4). Il est donc important de pouvoir évaluer précisément des déterminants. Or on sait qu'un déterminant est un polynôme (à plusieurs variables) en les coefficients de la matrice. On peut donc appliquer le même genre de technique que pour l'évaluation polynomiale.

À l'heure actuelle, il y a plusieurs possibilités pour calculer de façon précise un déterminant. On peut bien sûr utiliser des arithmétiques exactes mais cela est très coûteux en temps. La méthode considérée comme la plus efficace en temps est d'utiliser les expansions de Shewchuk [180]. Il s'agit d'une arithmétique adaptative qui permet d'augmenter la précision des calculs jusqu'à l'obtention de la précision désirée. Récemment, Demmel et Hida [52] ont proposé des algorithmes de tests de ces prédicats en utilisant un algorithme de sommation précise basé sur la présence d'accumulateurs larges dans l'architecture considérée. Le problème de ces algorithmes est qu'ils ne sont pas portables du fait du choix d'accumulateurs larges et de plus nécessitent de trier les entrées ce qui est souvent coûteux.

Contribution : Au lieu d'utiliser une sommation par accumulateurs larges, nous proposons d'utiliser l'algorithme récent de sommation précise d'Ogita, Rump et Oishi [153]. Leur algorithme permet aussi de calculer une borne certifiée sur l'erreur de la sommation. On connaît le signe d'un nombre si l'erreur relative commise lors de son calcul est inférieure à un. Nous transformons le calcul du déterminant en le calcul d'une somme. Nous utilisons alors l'algorithme de [153] afin de calculer la somme avec une précision relative inférieure à un. Les performances de notre algorithme sont de l'ordre de grandeur de celui de Shewchuk [180] dans les cas bien conditionnés mais il est deux fois plus rapides dans les cas mal conditionnés.

Ce travail en collaboration avec Nicolas Louvet fait l'objet de l'article [74] accepté à la conférence internationale REC 2006.

3.2 Partie II : Pseudozéros, conditionnement et applications

Étant donné $\varepsilon > 0$, l'ensemble des ε-pseudozéros d'un polynôme p est l'ensemble des zéros de tous les polynômes se trouvant à une distance (pour une norme donnée, voir début du chapitre 5) inférieure à ε de p. Il s'agit d'un outil introduit par Mosier [143] en 1986 afin d'étudier la sensibilité des zéros de polynômes par rapport à des perturbations sur les coefficients. Les pseudozéros ont ensuite été étudiés par Trefethen et Toh [192] qui ont comparé la notion de pseudozéros à celle de pseudospectres de la matrice compagnon. Plus récemment, Stetter [186] a consacré un chapitre de son livre à cette notion. Le chapitre 5 donne une présentation générale de la notion de pseudozéros.

3.2.1 Applications des pseudozéros en calcul formel (chapitre 6)

Motivation : On montre ici la limite de la définition du PGCD algébrique dans le domaine de la précision finie. Prenons par exemple deux polynômes p et q unitaires et tels que $\deg p > 1$. On suppose de plus que p divise q. Cela revient à dire que $\mathrm{PGCD}(p, q) = p$. Or pour toute constante $\varepsilon > 0$, on a $\mathrm{PGCD}(p, q + \varepsilon) = 1$. Par conséquent, une petite perturbation du polynôme p peut entraîner un « saut » important du PGCD. Le PGCD ne dépend donc pas continûment des perturbations de ses coefficients. Le calcul du PGCD est par conséquent un problème mal posé au sens d'Hadamard.

On a le même type de problèmes avec la notion de polynômes « premiers entre eux ». L'exemple suivant [12] est révélateur. Soient p et q les polynômes suivants

$$p(z) = (z - \frac{1}{3})(z - \frac{5}{3}) = z^2 - 2z + \frac{5}{9}, \quad q(z) = z - \frac{1}{3}.$$

Lorsque l'on transforme les coefficients de p et q en nombres flottants, les deux polynômes, qui ont une racine commune en arithmétique exacte, deviennent premiers entre eux. De la même façon, les polynômes

$$p(z) = 50z - 7, \quad q(z) = z - \frac{1}{7},$$

premiers entre eux en arithmétique exacte, ne le sont plus en considérant une précision de deux chiffres décimaux après la virgule en base 10 avec arrondi au plus proche ($1/7 = 0.14285714$ et $7/50 = 0.14$).

Contribution : Nous introduisons le problème du calcul du PGCD approché de deux polynômes lorsque les coefficients sont des nombres complexes ou réels approchés. Un PGCD approché (ou ε-PGCD) est défini comme suit :

Étant donnés deux polynômes p et q de degrés respectifs n et m, et ε un réel strictement positif, on appelle *ε-diviseur* (ou *diviseur approché*) de p et q tout diviseur de polynômes perturbés $\widehat{p}$ et $\widehat{q}$ vérifiant $\|p - \widehat{p}\| \leqslant \varepsilon$, $\|q - \widehat{q}\| \leqslant \varepsilon$ et $\deg(p - \widehat{p}) \leqslant n$, $\deg(q - \widehat{q}) \leqslant m$. Un ε-PGCD de p et q est un *ε-diviseur* de degré maximum.

Nous présentons succinctement les différentes méthodes existantes pour calculer un PGCD approché. Nous distinguons les méthodes à base d'optimisation, de divisions euclidiennes et de méthodes matricielles (SVD). Nous proposons ensuite une étude géométrique du PGCD approché grâce à l'utilisation des pseudozéros. Pour finir, nous regardons la primalité de deux polynômes à coefficients approchés. Il est clair qu'un calcul de PGCD approché permet de répondre à la question. Néanmoins, d'autres méthodes semblent plus rapides. Nous étudions d'abord un algorithme proposé par Beckermann et Labahn [11, 12], puis nous montrons que les tracés de pseudozéros permettent aussi de répondre simplement à cette question.

Ce travail en collaboration avec Philippe Langlois fait l'objet du rapport de recherche [72]. Il est actuellement en cours de révision à Theoretical Computer Science.

3.2.2 Applications des pseudozéros en théorie du contrôle (chapitre 7)

Motivation : En théorie du contrôle et en automatique, on écrit classiquement les fonctions de transfert sous la forme $H(p) = N(p)/D(p)$, où N et D sont deux polynômes et où p est le paramètre du système. Le système décrit par cette fonction de transfert est dit *stable* (dans le sens de Hurwitz) si tous les zéros de D sont à partie réelle négative (autrement dit si le polynôme D est stable au sens de Hurwitz). Puisque des incertitudes sur les coefficients sont inévitables dans les problèmes de la vie réelle (incertitudes sur les données, erreurs d'arrondi), il est utile de mesurer la distance d'un système stable au système instable le plus proche. Il s'agit en fait de mesurer la distance de D au polynôme instable le plus proche. Cette distance s'appelle le rayon de stabilité de D.

Contribution : Nous proposons un algorithme de calcul du rayon de stabilité. Le point clé de notre algorithme est l'utilisation des pseudozéros. L'algorithme proposé est de type symbolique-numérique (on voit aussi le terme d'algorithme hybride). Cela signifie que l'on utilise des méthodes formelles (ici les suites de Sturm) dans des procédures numériques. Une telle approche a été initiée dans [22] pour compter le nombre de valeurs propres imaginaires pures d'une matrice Hamiltonienne. Il semble néanmoins que de telles approches aient été très peu étudiées bien qu'elles fournissent des algorithmes efficaces et précis.

Ce travail en collaboration avec Philippe Langlois fait l'objet du rapport de recherche [71] qui sera soumis à IEEE Transactions on Automatic Control.

3.2.3 Zéros de polynômes d'intervalles (chapitre 8)

Motivation : Le concept de l'analyse par intervalles [107] est de calculer avec des intervalles de nombres réels plutôt qu'avec les nombres réels eux-mêmes. Tandis que l'arithmétique flottante est affectée par les erreurs d'arrondis et donc peut conduire à des résultats imprécis et faux, l'analyse par intervalles a l'avantage de donner des bornes rigoureuses pour la solution exacte. Une telle analyse se révèle très utile quand par exemple les paramètres ne sont connus qu'avec une certaine incertitude. Dans ce cas, on peut implémenter des algorithmes en utilisant l'arithmétique d'intervalles pour les paramètres incertains afin de produire un intervalle qui contient tous les résultats possibles.

Contribution : Nous nous intéressons à la notion de polynômes d'intervalles. Il s'agit de polynômes dont les coefficients ne sont plus des nombres réels ou complexes mais des intervalles réels. Cela va nous permettre de modéliser des incertitudes sur les coefficients d'un polynôme. Nous nous intéressons aux pseudozéros réels de polynômes réels avec des perturbations mesurées en norme infinie. Cela nous permet de donner une formule calculable et un outil pour tracer les pseudozéros d'un polynôme d'intervalles.

Ce travail fait l'objet de l'article [73] en commun avec Philippe Langlois à paraître dans les actes de la conférence internationale ACM SAC 2006.

3.2.4 Pseudozéros de polynômes à plusieurs indéterminées (chapitre 9)

Motivation : Les polynômes à plusieurs variables apparaissent dans presque tous les champs du calcul scientifique et de l'ingénierie comme on peut le voir en lisant le « Computer Algebra Handbook » [68], dans [53] et dans le rapport du projet européen FRISCO. La très large utilisation des systèmes polynomiaux nécessite d'avoir des algorithmes de résolution stables et efficaces. Actuellement il y a principalement deux grandes approches de résolution : l'une symbolique et l'autre numérique. L'approche symbolique est soit basée sur la théorie des bases de Gröbner soit sur la théorie des résultants. Pour l'approche numérique, on utilise plutôt des méthodes itératives telles que la méthode de Newton (souvent des versions améliorées) ou bien des méthodes d'homotopie. Récemment, des méthodes hybrides, combinant à la fois des méthodes symboliques et numériques sont apparues (voir le chapitre « Hybrid Methods » de Kaltofen dans [68, p.112-128]).

Contribution : La majeure partie des articles sur les pseudozéros considèrent seulement les pseudozéros de polynômes à une indéterminée. Les cas avec plusieurs indéterminées semblent avoir été très peu étudiés excepté par Stetter [185, 186], par Hoffman, Madden et Zhang [98] et Corless, Kai et Watt [40]. De plus, le cas à plusieurs indéterminées a seulement été traité dans le cas de polynômes (et de systèmes polynomiaux) à coefficients complexes. Nous considérons ici des systèmes où les polynômes sont à coefficients réels et tels que tous les polynômes de tous les systèmes perturbés aient aussi des coefficients

réels. Nous explicitons une formule calculable pour tracer l'ensemble des pseudozéros et nous étudions différentes méthodes pour les visualiser.

Ce travail a fait l'objet d'un poster à la conférence internationale CASC 2005.

3.2.5 Conditionnement structuré de racines de polynômes (chapitre 10)

Motivation : Le conditionnement de racines de polynômes consiste à mesurer la variation de la racine si l'on perturbe de manière infinitésimale les coefficients du polynôme. Ce conditionnement a été étudié pour des polynômes à coefficients complexes et où l'on autorise des perturbations complexes de ces coefficients (voir par exemple [36, 206]). Il en est de même pour l'erreur inverse.

Contribution : Comme pour l'évaluation polynomiale (chapitre 2), les polynômes à considérer sont souvent à coefficients réels. Les perturbations que peuvent subir ces coefficients sont en général elles aussi réelles. Nous avons défini un nouveau conditionnement pour les polynômes à coefficients réels en autorisant seulement des perturbations réelles des coefficients. Nous donnons une formule calculable pour ce *conditionnement réel*. Nous montrons surtout que le rapport entre ce conditionnement réel et le conditionnement classique se trouve dans l'intervalle $[1, \sqrt{2}]$. Par conséquent, le conditionnement classique est suffisant pour rendre compte de la difficulté de calculer des racines. Nous avons aussi défini une erreur inverse réelle. Nous donnons aussi une formule calculable pour cette erreur inverse réelle. Nous montrons qu'elle peut être très différente de l'erreur inverse classique.

3.3 Partie III : Pseudospectre et conditionnement matriciel

Motivation : Ce passage est tiré de [172] et illustre la nécessité de prendre en compte la structure de certains problèmes matriciels. Considérons un problème numérique à m paramètres d'entrée produisant k paramètres en sortie. Il s'agit en fait d'une fonction $f : \mathbf{R}^m \to \mathbf{R}^k$. Un algorithme de résolution du problème, *i.e.,* de calcul de f, peut être considéré comme une fonction $\widetilde{f}$. Pour une entrée $p \in \mathbf{R}^m$, le résultat numérique $\widetilde{f}(p)$ sera en général identique pour tout $\widetilde{p}$ dans un petit voisinage autour de p. Par conséquent, on ne peut espérer plus d'un algorithme numérique qu'il produise la valeur exacte $f(\widetilde{p})$ pour un $\widetilde{p}$ proche de p. Un algorithme avec cette propriété est dit *inverse-stable*. Par exemple, la méthode de Gauss avec pivotage partiel pour résoudre des systèmes linéaire est inverse-stable.

Mais est-il toujours possible que $\widetilde{f}(p) = f(\widetilde{p})$ pour un $\widetilde{p}$ proche de p ? Pour ce faire, considérons le calcul en double précision IEEE 754 du carré d'une matrice, par exemple $A = \begin{pmatrix} 1+u & 4 \\ 4 & -1 \end{pmatrix}$, où $u = 2^{-52}$, et tel que 1 et $1+u$ soient des nombres flottants adjacents.

Le résultat est $\widetilde{B} = \mathrm{fl}(A^2) = \begin{pmatrix} 17 & 4u \\ 4u & 17 \end{pmatrix}$. Pour une perturbation $\Delta A = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}$, on a

$$(A+\Delta A)^2 = \begin{pmatrix} (1+u+\alpha)^2 + (4+\beta)(4+\gamma) & (4+\beta)(u+\alpha+\delta) \\ (4+\gamma)(u+\alpha+\delta) & (4+\beta)(4+\gamma) + (1-\delta)^2 \end{pmatrix}.$$

Mais $(A+\Delta A)^2 = \widetilde{B}$ est impossible pour des petites perturbations ΔA car cela implique en comparant $\widetilde{B}_{11}$ et $\widetilde{B}_{22}$ que $(1+u+\alpha)^2 = (1-\delta)^2$ et donc que $u+\alpha = -\delta$. Mais alors on a $(A+\Delta A)_{12} = 0$. Autrement dit, la multiplication matricielle donne la meilleure approximation en double précision $\widetilde{B}$ du résultat exact A^2 mais n'est pas inverse-stable. Un tel comportement est fréquent pour les problèmes structurés.

Considérons par exemple un système linéaire $Cx = b$ avec C une matrice circulante. Beaucoup d'algorithmes tirent parti de telles informations que ce soit en terme de temps de calcul et de taille de stockage. Dans notre cas, la première ligne de la matrice et le second membre sont les entrées du solveur structuré et donc $m = 2n$ entrées sont envoyées en $k = n$ sorties. Par nature, une perturbation de la matrice C est une matrice circulante.

On peut trouver des exemples de systèmes $Cx = b$ tels que pour une solution calculée $\widetilde{x}$ il est probable que $(C+\Delta C)\widetilde{x} \neq b + \Delta b$ pour des perturbations ΔC et Δb telles que $C + \Delta C$ soit circulante alors même que $\widetilde{x}$ est très proche de la solution exacte de $Cx = b$. L'explication provient du fait que contrairement aux systèmes linéaires généraux, l'espace des entrées n'est pas assez riche pour produire des perturbations ayant la propriété désirée.

Dans un tel cas, tout ce qu'on peut attendre d'un algorithme en précision finie est qu'il produise un résultat $\widetilde{x}$ tel que $(C+\Delta C)(\widetilde{x}+\Delta x) = b + \Delta b$. Dans le formalisme précédent, cela signifie que pour une entrée p, l'algorithme $\widetilde{f}$ produit un résultat $q = \widetilde{f}(p)$ avec $q + \Delta q = f(p + \Delta p)$. Un algorithme $\widetilde{f}$ avec cette propriété est dit *stable*. Il existe en effet des algorithmes stables pour résoudre des systèmes linéaires avec des matrices circulantes. Tout cela motive une analyse des perturbations structurées, des conditionnements structurés et des erreurs inverses structurées.

Les perturbations structurées ne sont pas aussi simples à traiter que les perturbations non structurées et l'analyse de l'erreur inverse est souvent plus difficile quand cela concerne des perturbations structurées.

3.3.1 Pseudospectres et structuration (chapitre 11)

Contribution : La notion de pseudospectre, notion analogue aux pseudozéros, a été popularisée par Trefethen [193]. Il s'agit de l'ensemble des valeurs propres des matrices proches d'une matrice donnée. Nous proposons une notion de pseudospectre structuré, c'est-à-dire que nous n'autorisons que des perturbations structurées de la matrice. Nous montrons que pour les matrices Toeplitz, circulantes, Hankel et symétriques, les pseudospectres structurés et non structurés coïncident. Nous montrons que cela est faux pour les matrices hermitiennes et anti-hermitiennes. Nous généralisons ce type de résultats aux cas des matrices polynomiales. En effet, nous montrons que les pseudospectres structurés et non structurés coïncident pour les matrices polynomiales ayant les structures Toeplitz, circulantes, Hankel ou symétriques. Nous donnons aussi une formule calculable pour le

pseudospectre structuré de matrices polynomiales réelles. Ce type de matrices apparaît fréquemment en théorie du contrôle.

Ce travail fait l'objet de l'article [69] à paraître dans Journal of Computational and Applied Mathematics.

3.3.2 Conditionnement structuré de problèmes matriciels (chapitre 12)

Contribution : Nous nous intéressons aux effets des perturbations structurées sur la solution d'un système linéaire, de l'inversion de matrices et sur la distance à la singularité. Une attention particulière est donnée aux structures linéaires et non-linéaires que forment les algèbres de Lie, les algèbres de Jordan et le groupe d'automorphisme d'un produit scalaire. Cela inclut par exemple les matrices symétriques, pseudosymétriques, persymétriques, anti-symétriques, Hamiltoniennes, unitaires, orthogonales et symplectiques.

Nous montrons que sous des hypothèses souples sur le produit scalaire, il n'y a pas ou très peu de différences entre le conditionnement structuré et non structuré et entre la distance structurée et non structurée à la singularité pour les matrices appartenant à une algèbre de Lie ou de Jordan. Ainsi, pour ces classes de matrices, l'analyse usuelle pour des perturbations non structurées est suffisante. Nous montrons que ce n'est pas vrai en général pour des structures provenant d'un groupe d'automorphisme. Nous donnons des bornes et des expressions calculables pour le conditionnement structuré de systèmes linéaires et pour l'inversion de matrices pour ces structures non linéaires. Nous considérons aussi l'erreur inverse structurée pour les solutions approchées de systèmes linéaires. Nous donnons des conditions pour que l'erreur inverse soit finie. Nous montrons que pour les algèbres de Lie et de Jordan, l'erreur inverse (quand elle existe) se situe dans un petit facteur de l'erreur inverse non structurée.

Ce travail en collaboration avec Françoise Tisseur fait l'objet du rapport de recherche [189] . Il est soumis à Electronic Journal of Linear Algebra.

Chapitre 1

Introduction à l'analyse d'erreur en précision finie

Ce chapitre se veut une introduction à l'analyse d'erreur en précision finie. Il est très largement inspiré de [87, 121, 122, 36, 186]. Nous commencerons par une brève définition de ce qu'est l'analyse d'erreur et nous définirons les outils de base dont nous aurons besoin par la suite. Dans la section 1.1, nous définissons le conditionnement d'un problème. Dans la section 1.2, nous définissons la notion de stabilité d'un algorithme et en particulier la notion d'erreur directe et d'erreur inverse. Dans la section 1.3, nous montrons comment les outils des sections précédentes (conditionnement et erreur inverse) permettent d'obtenir des informations sur la précision de la solution. La section 1.4 concerne le modèle de l'arithmétique flottante et en particulier celui défini par la norme IEEE 754. La dernière section est consacrée à l'étude de méthodes permettant d'améliorer la précision des résultats.

Soit $\widehat{x}$ une approximation d'un nombre réel x non nul. Les deux principales façons de mesurer la précision de $\widehat{x}$ sont l'*erreur absolue*

$$E_a(\widehat{x}) = |x - \widehat{x}|,$$

et l'*erreur relative*

$$E_r(\widehat{x}) = \frac{|x - \widehat{x}|}{|x|}.$$

Soit Δx une perturbation absolue de $x \neq 0$ et $x + \Delta x$ la valeur perturbée. On notera $\|\Delta x\|$ une norme *relative* de la perturbation Δx (on a donc une dépendance implicite entre $\|\Delta x\|$ et x). Par conséquent, pour un élément x scalaire non nul, on a $\|\Delta x\| = |\Delta x|/|x|$. Lorsque x et Δx sont vectoriels (et $\|x\| \neq 0$), la norme $\|\Delta x\|$ dépend de la structure et de la norme vectorielle considérée. On distingue alors principalement les normes globales (*normwise*) $\|\Delta x\| = \|\Delta x\|^a/\|x\|^a$, où $\|\cdot\|^a$ est une norme (absolue) vectorielle et la norme par composante (*componentwise*) définie par $\|\Delta x\| = \max_i |\Delta x_i|/|x_i|$ $(x = (x_i), x_i \neq 0)$.

La précision d'une solution calculée par un algorithme dépend intimement du conditionnement du problème et de la stabilité de l'algorithme utilisé. Le conditionnement mesure l'effet des perturbations des données sur la solution. L'erreur inverse modélise

les erreurs introduites par l'algorithme exécuté en précision finie comme des erreurs sur les données. Dans ce cadre, la précision vérifie au premier ordre la célèbre estimation empirique

$$\text{précision} \lesssim \text{conditionnement} \times \text{erreur inverse}. \tag{1.1}$$

Notre objectif dans ce chapitre est d'étudier la qualité d'une approximation. Dans ce but, il convient de distinguer deux parties fondamentales distinctes :

– la difficulté de résoudre le problème ;
– la stabilité de l'algorithme de résolution de ce problème.

Les deux prochaines sections traitent de chacun de ces aspects.

1.1 Conditionnement d'un problème

1.1.1 Problème bien posé

Considérons le problème mathématique P suivant

$$(P) : \text{trouver } x \text{ tel que } F(x) = y \text{ pour } y \text{ donné},$$

où F est une application continue, non linéaire en général, entre des espaces vectoriels normés qui sont souvent de la forme $\mathbf{R}^n$ ou $\mathbf{C}^n$. On dira que le problème (P) est *bien posé* (au sens d'Hadamard) si la solution $x = F^{-1}(y)$ existe, est unique et dépend continûment des données y (que l'on peut remplacer par le fait que F^{-1} soit continue au voisinage de y). Si ce n'est pas le cas, le problème est dit *mal posé*. Nous n'étudierons dans la suite de ce chapitre que des problèmes bien posés. Des exemples de problèmes bien posés sont par exemple l'évaluation polynomiale ou bien la résolution d'un système linéaire $Ax = b$ où A est bien sûr inversible. Par contre la résolution d'un système linéaire $Ax = b$ où A est singulière est un exemple de problème mal posé.

1.1.2 Conditionnement d'un problème

En supposant que le problème (P) est bien posé, nous pouvons le réécrire sous la forme

$$x = G(y),$$

où $G = F^{-1}$. Nous supposerons ici que G est différentiable. Si l'on commet une erreur Δy sur la donnée y, la valeur calculée sera $\widehat{x} = G(y + \Delta y)$. L'erreur Δy sera souvent appelée la *perturbation* de y. Si Δy est suffisamment petite, on a au premier ordre l'approximation

$$\widehat{x} - x \approx G'(y) \cdot \Delta y,$$

ce que l'on peut écrire sous la forme (si $x \neq 0$ et $y \neq 0$)

$$(\widehat{x} - x)/x \approx (yG'(y)/G(y)) \cdot (\Delta y/y).$$

Ces relations expriment respectivement l'effet absolu et relatif sur le résultat $\widehat{x}$ de la variation absolue ou relative de la donnée y, en ne prenant en compte que les effets d'ordre 0 et 1. On en déduit la relation suivante

$$\frac{|\widehat{x}-x|}{|x|} = K(G,y)\cdot\frac{|\Delta y|}{|y|} + \mathcal{O}(|\Delta y|^2)$$

où

$$K(G,y) = \left|\frac{yG'(y)}{G(y)}\right|. \tag{1.2}$$

Le coefficient $K(G,y)$ est le nombre de conditionnement relatif de l'évaluation de G en y.

Exemple (évaluation polynomiale 1). Considérons l'évaluation d'un polynôme $p(z) = \sum_{i=0}^{n} a_i z^i$, $a_i, z \in \mathbf{R}$, en perturbant la seule donnée $y = z$. Si $z \neq 0$, la relation (1.2) fournit $K(p,z) = |zp'(z)/p(z)|$. On remarque facilement que le conditionnement tend vers l'infini quand la donnée z se rapproche d'une racine de p. Ainsi la précision de l'évaluation d'un polynôme au voisinage d'une de ses racines peut être arbitrairement mauvaise.

On va maintenant analyser la sensibilité du problème (P) afin de quantifier l'effet de perturbations infinitésimales de la donnée y sur la solution x. Les espaces de données et de résultats sont notés respectivement $\mathcal{D}$ et $\mathcal{R}$; on notera $\|\cdot\|_{\mathcal{D}}$ et $\|\cdot\|_{\mathcal{R}}$ des normes relatives sur ces espaces. Étant donnés $\varepsilon > 0$ et $\mathcal{P}(\varepsilon) \subset \mathcal{D}$ un ensemble de perturbations Δy de la donnée y qui vérifient $\|\Delta y\|_{\mathcal{D}} \leqslant \varepsilon$, le problème perturbé associé au problème (P) est défini par

$$\text{trouver } \Delta x \in \mathcal{R} \text{ tel que } F(x+\Delta x) = y + \Delta y \text{ pour } \Delta y \in \mathcal{P}(\varepsilon) \text{ donné.} \tag{1.3}$$

On suppose aussi que x et y sont non nuls. Le *nombre de conditionnement* du problème (P) en la donnée y est défini par

$$K(P,y) := \lim_{\varepsilon\to 0}\sup_{\Delta y\in\mathcal{P}(\varepsilon)}\left\{\frac{\|\Delta x\|_{\mathcal{R}}}{\|\Delta y\|_{\mathcal{D}}}\right\}.$$

Exemple (évaluation polynomiale 2). Considérons maintenant le problème de l'évaluation polynomiale en prenant en compte les perturbations sur les coefficients du polynôme. Soit $p(z) = \sum_{i=0}^{n} a_i z^i$, avec $a_i, z \in \mathbf{R}$ et $a_i \neq 0$, et $\Delta y = (\Delta a_0, \ldots, \Delta a_n)^T = \Delta a$ une perturbation de la donnée $y = (a_0, \ldots, a_n)^T = a$. Le résultat $x = p(y)$ est perturbé en $\widehat{x} = p(y+\Delta y) = \sum_{i=0}^{n}(a_i + \Delta a_i)z^i$. La perturbation est donc $\Delta x = \widehat{x} - x = \sum_{i=0}^{n} \Delta a_i z^i$. On munit $\mathcal{D} = \mathbf{R}^{n+1}$ de la norme relative $\|\Delta a\|_{\mathcal{D}} = \max_{i=0:n} |\Delta a_i|/|a_i|$ et $\mathcal{R} = \mathbf{R}$ de la norme relative $\|\Delta x\|_{\mathcal{R}} = |\Delta x|/|x|$. Puisque $|\Delta x| = |\sum_{i=0}^{n} \Delta a_i z^i| \leqslant \|\Delta a\|_{\mathcal{D}} \sum_{i=0}^{n} |a_i||z|^i$, on a

$$\frac{\|\Delta x\|_{\mathcal{R}}}{\|\Delta y\|_{\mathcal{D}}} \leqslant \frac{\sum_{i=0}^{n}|a_i||z|^i}{|\sum_{i=0}^{n} a_i z^i|}.$$

Cette borne est atteinte pour la perturbation Δa telle que $\Delta a_i/a_i = \text{sign}(a_i z^i)\|\Delta a\|_{\mathcal{D}}$. Le nombre de conditionnement relatif pour l'évaluation de p en z est

$$K(p(z), a) = \frac{\sum_{i=0}^{n} |a_i z^i|}{|p(z)|}.$$

On remarque que ce nombre est d'autant plus grand que l'on évalue le polynôme au voisinage d'une de ses racines.

L'exemple précédent montre que calculer un nombre de conditionnement peut ne pas être très facile (on a dû trouver une majoration et ensuite un terme qui atteint la borne, ce qui n'est pas très aisée en général). On va montrer maintenant que lorsque la fonction G du problème (P) est de classe C^1 alors le conditionnement est intimement lié à la dérivée de G. On considère ici des normes globales relatives $\|\Delta z\|/\|z\|$ sur les espaces $\mathcal{D}$ et $\mathcal{R}$. On notera par $|||\cdot|||$ la norme subordonnée sur l'espace $\mathscr{L}(\mathcal{D}, \mathcal{R})$ des applications linéaires de $\mathcal{D}$ dans $\mathcal{R}$. La solution du problème perturbé (1.3) vérifie $x+\Delta x = G(y+\Delta y)$. En développant au premier ordre, on peut écrire $\Delta x = G(y + \Delta y) - G(y) = G'(y)\Delta y + \mathcal{O}(\|\Delta y\|_{\mathcal{D}}^2)$. On en déduit alors que

$$K(P, y) = |||G'(y)||| \, \frac{\|y\|_{\mathcal{D}}}{\|G(y)\|_{\mathcal{R}}},$$

et pour les normes absolues

$$K_a(P, y) = |||G'(y)||| \, .$$

Le conditionnement est donc à un facteur près la norme de la dérivée de l'application G.

Maintenant que nous avons donné quelques formules pour calculer un nombre de conditionnement, il nous reste à savoir comment interpréter ce nombre. On dit qu'un problème est *mal conditionné* s'il admet un nombre de conditionnement élevé ; sinon on dit qu'il est *bien conditionné*. Cependant il n'est pas facile de donner une frontière simple entre problèmes bien et mal conditionnés. Le conditionnement mesure la sensibilité du résultat pour des perturbations des données. Plus le conditionnement est grand et plus une petite perturbation sur les données peut entraîner une erreur importante sur le résultat. Néanmoins, il ne faut pas oublier que le conditionnement mesure le « pire » effet possible des perturbations sur les données. Il se peut donc qu'un problème soit mal conditionné et que les perturbations soient telles qu'en fait il y a très peu d'erreurs sur le résultat.

Exemple. Considérons le système linéaire $Ax = b$ où A est une matrice inversible. Avec les mêmes notations que pour le problème (P) on a $x = G(A) = A^{-1}b$. On peut alors montrer [87, p.110] que $K(P, A) = \|A\|\|A^{-1}\|$ pour des normes vectorielles et subordonnées relatives. Le nombre $\kappa(A) = \|A\|\|A^{-1}\|$ est abusivement appelé le nombre de conditionnement de la matrice A, alors que le conditionnement est associé à un problème.

Avant de terminer cette section sur le conditionnement, nous allons faire quelques remarques importantes sur le sujet.

Remarques.
- Pour un même problème, on peut définir beaucoup de nombres de conditionnement différents en fonction des normes choisies sur les espaces de données et de résultats ainsi que sur le type de perturbations $\mathcal{P}(\varepsilon)$ choisi.

– Le nombre de conditionnement correspond en général à l'inverse de la distance au problème singulier le plus proche. Sous cette hypothèse, un problème mal conditionné est un problème proche d'un problème singulier. On peut montrer par exemple [87, p. 111] que

$$1/\kappa(A) = \min\{\|\Delta A\|/\|A\| : A + \Delta A \text{ singulière}\},$$

et

$$\frac{|p(x)|}{\sum_{i=0}^{n} |a_i x^i|} = \min\{d(p,q) : q(x) = 0\},$$

où $d(p,q) = \min\{|a_i - b_i|/|a_i|\}$ avec $q(x) = \sum_{i=0}^{n} b_i x^i$.

– Le nombre de conditionnement est indépendant de tout algorithme de résolution du problème. C'est donc une caractéristique intrinsèque du problème.

1.2 Stabilité d'un algorithme

Le conditionnement quantifie les (pires) effets des erreurs sur les données en ne prenant en compte que la nature du problème lui-même. Le conditionnement mesure donc la difficulté de résoudre un problème indépendamment de la façon de le résoudre.

On résout généralement un problème en utilisant un *algorithme*. Il s'agit d'un ensemble d'opérations et de tests que l'on peut en fait confondre avec la fonction G donnant la solution de notre problème. À cause d'un certain nombre d'erreurs de calcul et d'approximations (dues par exemple au fait que l'on ne peut pas représenter l'ensemble des nombres réels sur un ordinateur), l'algorithme n'est pas la fonction G mais une fonction $\widehat{G}$. L'algorithme ne calcule donc par $x = G(y)$ mais $\widehat{x} = \widehat{G}(y)$.

L'*analyse directe* étudie la nouvelle fonction $\widehat{G}$ de façon à pouvoir estimer l'écart entre x et $\widehat{x}$. Le calcul de $\widehat{G}$ est en général assez difficile car il faut suivre de manière précise la propagation des erreurs tout au long des calculs. L'écart entre la solution exacte x et la solution calculée $\widehat{x}$ est appelé l'*erreur directe*. Elle correspond en fait à la précision de la solution calculée. On ne cherche en général qu'une majoration de l'écart entre x et $\widehat{x}$.

La difficulté de l'analyse directe réside dans le calcul de $\widehat{G}$. L'*analyse inverse* permet de contourner ce problème en travaillant directement sur la fonction G elle-même. L'idée est de chercher quel est le problème que l'on a effectivement résolu et si ce problème est « proche » (dans un sens à définir) du problème initial. On cherche alors à ramener l'erreur sur le résultat en une erreur sur la donnée. On cherche donc Δy tel que $\widehat{x} = G(y + \Delta y)$. On dit alors que Δy est l'*erreur inverse* associée à $\widehat{x}$. On cherche Δy dans l'ensemble des données $\mathcal{D}$. Dans la pratique, sachant que l'on connaît $\widehat{x}$ et G, on arrive très souvent à obtenir une bonne majoration de Δy (en général plus facilement que pour l'erreur directe). La figure 1.1 récapitule le principe de l'analyse directe et de l'analyse inverse.

La *stabilité* d'un algorithme décrit l'influence des calculs en précision finie sur la qualité du résultat. Soit $\widehat{x} = \widehat{G}(y)$ la solution de notre problème (P) calculé par un algorithme exécuté en précision finie. L'*erreur inverse* associée à $\widehat{x} = \widehat{G}(y)$ est le scalaire $\eta(\widehat{x})$ défini, s'il existe, par

$$\eta(\widehat{x}) = \min_{\Delta y \in \mathcal{D}} \{\|\Delta y\|_{\mathcal{D}} : \widehat{x} = G(y + \Delta y)\}.$$

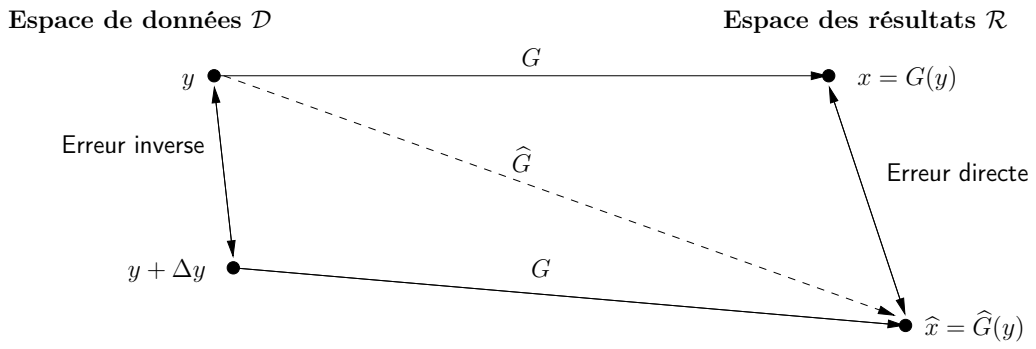


FIG. 1.1 – Erreurs directes et inverses lors du calcul de $x = G(y)$.

Si on ne peut trouver un tel Δy, on dit que $\eta(\widehat{x}) = +\infty$. Un algorithme est dit *inverse-stable* pour le problème (P) si la solution calculée $\widehat{x}$ a une erreur inverse $\eta(\widehat{x})$ « assez petite ». Un algorithme inverse-stable fournit donc la solution exacte d'un problème proche.

Exemple (évaluation polynomiale 3). Nous supposons dans cet exemple que les opérations d'addition et de multiplication satisfont la propriété

$$\widehat{z} = z(1+\delta) = (x \circ y)(1+\delta), \quad \circ \in \{+, \cdot\}, \quad \text{avec } |\delta| \leqslant \mathbf{u}. \tag{1.4}$$

On s'intéresse à l'algorithme de Horner pour l'évaluation d'un polynôme $p(z) = \sum_{i=0}^{n} a_i z^i$.

Algorithme 1.1. Évaluation polynomiale par le schéma de Horner

$s_n = a_n$
for $i = n-1 : -1 : 0$
  $s_i = s_{i+1} \cdot z + a_i$
end
$\widehat{p}(z) = s_0$

Au vu de la relation (1.4), nous pouvons écrire que

$$s_i = ((z \cdot s_{i+1})(1+\delta_{i_1}) + a_i)(1+\delta_{i_2}), \quad \text{avec } |\delta_{i_1}|, |\delta_{i_2}| \leqslant \mathbf{u},$$

où $\mathbf{u}$ est supposé très petit. Pour simplifier, on peut écrire $1 + \theta_n = \prod_{i=1}^{n}(1+\delta_i)$, pour $|\delta_i| \leqslant \mathbf{u}$. En itérant l'équation précédente, on trouve

$$\widehat{p}(z) = a_0(1+\theta_1) + a_1(1+\theta_3)z + a_2(1+\theta_5)z^2 + \cdots \\ + a_{n-1}(1+\theta_{2n-1})z^{n-1} + a_n(1+\theta_{2n})z^n.$$

On peut donc interpréter l'évaluation calculée $\widehat{p}(z)$ comme l'évaluation exacte du polynôme $q(z) = \sum_{i=0}^{n} b_i z^i$ avec $b_i = a_i(1+\theta_{2i+1})$, $i = 0 : n-1$, $b_n = a_n(1+\theta_{2n})$ en le point z. Comme pour $|\delta_i| \leqslant \mathbf{u}$ et $n\mathbf{u} < 1$, on a $|\theta_n| \leqslant n\mathbf{u}/(1-n\mathbf{u})$, on en déduit que l'erreur inverse vérifie

$$\eta(\widehat{x}) = |\theta_{2n}| \leqslant 2n\mathbf{u}.$$

L'erreur inverse est petite pour $\mathbf{u}$ petit et par conséquent l'algorithme de Horner est inverse-stable.

L'étude de la stabilité inverse en précision finie se fait en comparant l'erreur inverse avec la précision $\mathbf{u}$ à laquelle se font les calculs. On dit qu'un algorithme qui calcule $\widehat{x}$ en précision $\mathbf{u}$ est inverse-stable si l'erreur inverse $\eta(\widehat{x})$ est de l'ordre de $\mathbf{u}$.

1.3 Précision de la solution

La précision de la solution calculée dépend du conditionnement du problème et de la stabilité de l'algorithme utilisé. Le conditionnement mesure l'effet des perturbations des données sur la solution. L'erreur inverse modélise les erreurs introduites par l'algorithme exécuté en précision finie comme des erreurs sur les données. Dans ce cadre, la précision vérifie au premier ordre l'estimation empirique (1.1). La précision $\|\Delta x\|$ d'une solution calculée par un algorithme inverse-stable vérifie la relation

$$\|\Delta x\| \lesssim K\mathbf{u}, \tag{1.5}$$

où, le nombre de conditionnement K, la stabilité inverse et la précision sont définis pour des perturbations et des normes homogènes. On déduit de la relation (1.5) une caractérisation habituellement admise de la séparation entre problèmes bien et mal conditionnés. On dira donc qu'un problème est mal conditionné en calcul à précision $\mathbf{u}$ si son nombre de conditionnement est supérieur à $1/\mathbf{u}$ soit si $K\mathbf{u} \gtrsim 1$.

1.4 Introduction à l'arithmétique flottante

Cette section s'inspire très largement de [87, 158, 65, 182, 103, 104]. Les nombres à virgule flottante sont le principal mode de représentation des nombres dans les calculateurs et les ordinateurs.

1.4.1 Les nombres flottants

Un nombre flottant normalisé x s'écrit sous la forme

$$x = (-1)^s \times m \times b^e, \tag{1.6}$$

où $(-1)^s$ représente le signe de x avec $s \in \{0, 1\}$, m est la mantisse en base b et e est l'exposant. La base b est un entier $\geqslant 2$ et l'exposant est un entier relatif vérifiant

$$e_{\min} \leqslant e \leqslant e_{\max}, \tag{1.7}$$

où $e_{\min}$ et $e_{\max}$ sont donnés et représentent les bornes sur l'exposant. Étant donné un entier positif p, une mantisse m à p chiffres en base b s'écrit sous la forme

$$m = x_0 b^0 + x_1 b^{-1} + \cdots + x_{p-1} b^{1-p}, \tag{1.8}$$

avec $x_0 \neq 0$ et $x_i \in \{0, 1, 2, \ldots, b-1\}$, $i = 0 : p-1$. Puisque $x_0 \neq 0$, il y a unicité de l'écriture. De même, on en déduit aussi que la mantisse m vérifie $1 \leqslant m < b$.

Nous noterons par $\mathbf{F}(b, p, e_{\min}, e_{\max})$ l'ensemble des nombres flottants x définis par les relations (1.6), (1.7) et (1.8) union le singleton $\{0\}$. On notera les flottants par la notation classique $x = \pm x_0.x_1 \dots x_{p-1} \times b^e$. On remarque que tout nombre flottant non nul vérifie

$$\lambda := b^{e_{\min}} \leqslant |x| \leqslant (1 - b^{-p}) b^{e_{\max}} =: \sigma.$$

Si l'on travaille en base $b = 2$, le premier bit de tout flottant normalisé vaut 1. Par conséquent, il n'est pas utile de le stocker. On peut donc ne stocker que la partie fractionnaire f. Ainsi tout flottant normalisé en base 2 s'écrira sous la forme

$$x = (-1)^s \times 1.f \times 2^e \quad \text{avec } e_{\min} \leqslant e \leqslant e_{\max}.$$

On définit la *précision machine* ε comme la distance de 1.0 au nombre flottant suivant le plus proche. On montre alors que $\varepsilon = b^{1-p}$.

Théorème 1.1. *L'espace entre un nombre flottant normalisé et son successeur est d'au moins $b^{-1}\varepsilon|x|$ et d'au plus $\varepsilon|x|$.*

Le fait que $x_0 \neq 0$ pour les flottants normalisés empêche de pouvoir coder des nombres plus petits que λ. Or il se peut que dans des applications nous ayons besoin de nombres plus petits. L'idée consiste à autoriser que x_0 puisse être nul. Les *nombres flottants dénormalisés* sont les nombres x qui s'écrivent sous la forme

$$x = (-1)^s \times 0.f \times b^{e_{\min}}.$$

On remarque que le plus petit nombre dénormalisé positif est $\lambda_d = \lambda\varepsilon$. Les dénormalisés pallient donc l'absence de flottants normalisés entre 0 et $|\lambda|$.

1.4.2 Le modèle de l'arithmétique flottante

L'utilisation des nombres flottants sur les ordinateurs nécessite d'approcher chaque nombre réel par un nombre flottant. Cela revient à définir une application $\mathrm{fl} : \mathbf{R} \to \mathbf{F}$, $x \mapsto \mathrm{fl}(x)$ qui a tout nombre réel associe un nombre flottant. Cette application définira un arrondi de $\mathbf{R}$ vers $\mathbf{F}$ si elle vérifie les propriétés suivantes :

(i) $\mathrm{fl}(x) = x$ pour tout $x \in \mathbf{F}$;

(ii) $\mathrm{fl}(x) \leqslant \mathrm{fl}(y)$ pour tous $x, y \in \mathbf{R}$ tels que $x \leqslant y$.

Il y a plusieurs façons de définir un arrondi $\mathrm{fl}(\cdot)$. On peut prendre l'arrondi au plus près, *i.e.*, $\mathrm{fl}(x) = \mathrm{argmin}_{y \in \mathbf{F}} |x - y|$ ou bien par exemple prendre des arrondis dirigés soit vers 0 soit vers $+\infty$ ou $-\infty$.

L'intervalle des nombres flottants normalisés positifs est $[\lambda, \sigma]$. Prenons $x \in \mathbf{R}$ un réel que nous voulons arrondir dans $\mathbf{F}$. Nous dirons qu'il y a un *overflow* si $|x| > \sigma$. Nous dirons qu'il y a un *underflow* si $|x| < \lambda$.

Lorsque l'on veut approcher $x \in \mathbf{R}$ par $\mathrm{fl}(x) \in \mathbf{F}$, il est intéressant de pouvoir quantifier l'erreur que l'on commet. C'est l'objet du théorème suivant. Il dit que l'erreur relative commise est inférieure à l'unité d'arrondi $\mathbf{u}$ qui dépend de la précision ε et du mode d'arrondi. On a $\mathbf{u} = \varepsilon/2$ pour l'arrondi au plus près et $\mathbf{u} = \varepsilon$ pour l'arrondi dirigé.

Théorème 1.2 (Higham [87, p.38]). *Soit $x \in \mathbf{R}$ tel que $\lambda \leqslant |x| \leqslant \sigma$ alors*

$$\mathrm{fl}(x) = x(1+\delta), \quad |\delta| \leqslant \mathbf{u}.$$

On utilisera des fois une version modifiée de ce théorème sous la forme suivante.

Théorème 1.3 (Higham [87, p.38]). *Soit $x \in \mathbf{R}$ tel que $\lambda \leqslant |x| \leqslant \sigma$ alors*

$$\mathrm{fl}(x) = x/(1+\delta'), \quad |\delta'| \leqslant \mathbf{u}.$$

Il nous faut maintenant essayer de munir l'ensemble $\mathbf{F}$ d'une arithmétique qui approche celle définie sur $\mathbf{R}$. Étant donnés une opération arithmétique $\circ \in \{+,-,\cdot,/\}$ et deux flottants $x, y \in \mathbf{F}$, le résultat de l'opération $x \circ y$ n'est pas toujours un flottant. Il semble naturel de demander que le résultat de l'opération $x \circ y$ soit l'arrondi dans $\mathbf{F}$ de $x \circ y$, c'est-à-dire $\mathrm{fl}(a \circ b)$. Le modèle classique utilisé dans l'analyse d'erreur décrit en partie ce comportement.

Modèle standard

$$\mathrm{fl}(x \circ y) = (x \circ y)(1+\delta), \quad |\delta| \leqslant \mathbf{u}, \quad \circ \in \{+,-,\cdot,/\}. \tag{1.9}$$

La modification de (1.9) suivante peut aussi être utile.

$$\mathrm{fl}(x \circ y) = \frac{x \circ y}{1+\delta'}, \quad |\delta'| \leqslant \mathbf{u}. \tag{1.10}$$

Le modèle (1.9) ignore les possibilités d'un underflow ou d'un overflow. Pour prendre en compte un possible underflow, on peut modifier le modèle de la façon suivante

$$\mathrm{fl}(x \circ y) = (x \circ y)(1+\delta) + \eta, \quad |\delta| \leqslant \mathbf{u}, \quad \eta \leqslant \lambda \mathbf{u}.$$

Nous avons toujours $\delta\eta = 0$. S'il y a underflow alors $\delta = 0$ sinon $\eta = 0$.

1.4.3 La norme IEEE 754

La norme IEEE 754 publiée en 1985 [103] définit une arithmétique flottante binaire. C'est le résultat du travail d'un groupe de travail du « IEEE Computer Society Computer Standards Committee ». Parmi les principes de la norme, on trouve le fait d'encourager les experts à développer des programmes numériques portables, rapides et robustes qui traitent les exceptions arithmétiques et qui permettent de développer les calculs de fonctions élémentaires et des arithmétiques de grandes précisions. La norme spécifie le format des nombres flottants, le résultat des opérations flottantes élémentaires et les comparaisons, les modes d'arrondi ainsi que les conversions entre les différents formats arithmétiques. La racine carrée est incluse dans les opérations de base. La norme ne dit rien sur les fonctions élémentaires telles que exp et cos. La norme définit deux formats de nombres flottants.

Type	Taille	Mantisse	Exposant	Unité d'arrondi	Intervalle
Simple	32 bits	23+1 bits	8 bits	$2^{-24} \approx 5,96 \times 10^{-8}$	$10^{\pm 38}$
Double	64 bits	52+1 bits	11 bits	$2^{-53} \approx 1,11 \times 10^{-16}$	$10^{\pm 308}$

Dans les deux formats, un bit est réservé pour le signe. Le standard spécifie que les opérations arithmétiques doivent donner un résultat comme si on calculait avec une précision infinie et que l'on arrondissait ensuite le résultat. La norme prévoit quatre types d'arrondi. L'arrondi par défaut est l'arrondi au plus près avec arrondi pair (zéro comme dernier bit) si l'on se situe au milieu de deux nombres flottants. L'arrondi vers $-\infty$ et $+\infty$ est aussi supporté par la norme ; cela facilite l'implémentation de l'arithmétique d'intervalle par exemple. Le quatrième mode d'arrondi est l'arrondi vers zéro (autrement dit la troncature).

La norme IEEE 754 fournit un système clos dans le sens où chaque opération produit un résultat. Elle définit pour ce faire des valeurs spéciales :

- les valeurs $+\infty$ et $-\infty$ représentent les infinis positifs et négatifs. Elles sont retournées comme résultat d'un overflow.
- les valeurs 0^+ et 0^- correspondent à l'inverse des valeurs infinies.
- la valeur NaN (Not a Number) correspond à des résultats indéterminés.

Type d'exception	Exemple	Résultats
Opérations invalides	$0/0, 0 \times \infty, \sqrt{-1}$	NaN (Not a Number)
Overflow		$\pm\infty$
Division par zéro	nombre fini/0	$\pm\infty$
Underflow		nombres dénormalisés
Inexacte	quand $\mathrm{fl}(x \circ y) \neq x \circ y$	arrondi correct

1.5 Améliorer la précision du résultat

Il y a plusieurs façons d'améliorer la précision d'un résultat. La plus naïve consiste à effectuer les mêmes calculs avec une arithmétique de précision supérieure.

1.5.1 Augmenter la précision du calcul

Une façon possible pour augmenter la précision est de travailler avec une précision courante plus grande. Cette précision supérieure peut être soit disponible en matériel soit émulée en logiciel.

Pour ce qui est du matériel, la norme IEEE 754 prévoit par exemple la double précision si le calcul initial avait été effectué en simple précision. Au-delà de la double précision, la norme IEEE prévoit une précision étendue mais elle n'est pas normalisée. Certains processeurs tel l'Intel Pentium possèdent des registres flottants de 80 bits et font donc systématiquement tous les calculs avec 80 bits indépendamment du type simple ou double des opérandes.

Des solutions logicielles existent : on parle alors de multiprécision. De nombreuses bibliothèques multiprécision ont été développées. On peut les diviser en trois grandes classes :

- les bibliothèques multiprécision utilisant un format *multiple-digit* où un nombre est exprimé comme une suite de bits couplé à un exposant. Les exemples de bibliothèques sont MPFUN [7, 8] de Bailey, MP [23] de Brent, MPFR [2] ou GMP [1].
- les bibliothèques à précision étendue fixée utilisant un format *multiple-component* mais avec un nombre limité de composants. Les exemples classiques sont les *double-double* [9] de Bailey, les *double double* [24] de Briggs (les nombres double-double sont représentés comme une somme non évaluée de deux doubles) et les *quad-double* [81] (les quad-double sont représentés comme une somme non évaluée de quatre doubles).
- les bibliothèques multiprécision arbitraires utilisant un format *multiple-component* où les nombres sont exprimés comme des sommes non évaluées d'un nombre arbitraire de nombres flottants. Des exemples de telles bibliothèques sont celles de Priest [164, 165] et Shewchuk [179, 180].

1.5.2 Réécrire les algorithmes

Une modification de l'ordre des instructions peut conduire à un résultat exact ou bien de meilleure qualité. Réécrire l'algorithme permet alors de réduire l'erreur finale. Des exemples typiques de ce type de résultat sont donnés par les algorithmes de sommation de n flottants. Lorsque que l'on veut additionner des nombres positifs, la sommation récursive ordonnée par ordre croissant est celle qui a la plus petit erreur directe a priori [87].

1.5.3 Correction des algorithmes

L'exemple de la sommation nous permet d'illustrer l'amélioration de la précision grâce à un processus de correction. Une telle approche est possible car nous pouvons dans ce cas calculer l'erreur que nous avons commise à chaque opération élémentaire. Rappelons par exemple la proposition suivante qui dit qu'avec une arithmétique flottante binaire avec arrondi au plus près, l'erreur commise lors de la sommation est un flottant et en plus qu'elle est calculable dans l'arithmétique flottante utilisée.

Proposition 1.1 (Dekker [46])**.** *Soient a et b deux nombres flottants dans $\mathbf{F}$ tels que $|a| > |b|$ et $s = \mathrm{fl}(a+b)$. On a alors*

$$a + b = s + \delta,$$

où δ est le nombre flottant de $\mathbf{F}$ calculé par

$$\delta = \mathrm{fl}((a - s) + b).$$

Kahan a utilisé ce résultat pour corriger l'algorithme de sommation de n nombres flottants. Après chaque sommation partielle $s_i = s_{i-1} + x_i$, son algorithme de sommation compensée calcule l'erreur associée δ_i et l'ajoute au terme suivant x_{i+1} avant de calculer la prochaine somme partielle s_{i+1}. Cet algorithme améliore la borne d'erreur d'un facteur n par rapport à l'algorithme de sommation récursif. De nombreuses variantes de ce type

de correction ont été proposées. Pour la sommation, la meilleure amélioration semble être l'algorithme doublement compensé de Priest [165]. Cet algorithme suppose que les entrées sont triées et renvoie un résultat quasi-optimal : en effet, la somme calculée $\widehat{s}$ satisfait $|\widehat{s} - s|/|s| \leqslant 2\mathbf{u}$.

Nous allons maintenant présenter une méthode de correction automatique [120] basée sur les mêmes types de techniques. Supposons que nous devons évaluer une fonction f en les données $D = (d_1, \ldots, d_n)$ avec un algorithme g utilisant des variables intermédiaires $d_{n+1}, \ldots, d_N$. Nous supposerons par simplicité que f et les d_i sont des quantités scalaires. Nous supposerons aussi que chaque d_k $(k > n)$ est le résultat d'une opération élémentaire, $+, -, \times, /$ ou une racine carrée. Nous noterons δ_k l'erreur d'arrondi associée. Au premier ordre, la solution calculée $\widehat{d_N}$ satisfait

$$\widehat{d_N} - f(d_1, \ldots, d_n) = \sum_{k=n+1}^{N} \frac{\partial g}{\partial d_k}(D, \delta)\delta_k. \tag{1.11}$$

Langlois [120] présente une méthode appelée CENA qui calcule la linéarisation (1.11) de l'erreur et additionne cette quantité à la solution calculée. Les dérivées sont calculées par différentiation automatique et les erreurs d'arrondi sont calculées exactement (avec les mêmes types d'outils que pour la proposition 1.1). Pour la classe des algorithmes linéaires définis dans [120] dont font partie par exemple l'algorithme de Horner, la relation (1.11) est exacte et la méthode CENA fournit une solution corrigée avec une faible erreur.

1.6 Conclusion

Dans ce chapitre, nous avons présenté les bases de l'analyse d'erreur. Nous nous sommes intéressés à la notion de conditionnement d'un problème, qui mesure la sensibilité de la solution à une perturbation des données. Nous nous sommes ensuite intéressés à la notion d'erreur inverse qui révèle ou non la stabilité d'un algorithme. Nous avons ensuite présenté les bases de l'arithmétique flottante et en particulier la norme IEEE 754. C'est la norme couramment implémentée pour les calculs en virgule flottante.

Toutes les notions présentées dans ce chapitre sont essentielles et seront utilisées fréquemment dans le reste de la thèse.

Première partie

Évaluation polynomiale

Chapitre 2

Conditionnement structuré de l'évaluation polynomiale

L'évaluation polynomiale est une opération constamment utilisées en calcul scientifique et en ingénierie. En pratique, les coefficients des polynômes ne sont connus qu'avec une certaine incertitude. La sensibilité de l'évaluation polynomiale par rapport à des perturbations sur les coefficients a été étudiée avec la notion de conditionnement (voir le chapitre 1). Le conditionnement d'un problème mesure la sensibilité du résultat par rapport à de petits changements sur les données.

Dans ce chapitre, nous nous intéressons au conditionnement de l'évaluation d'un polynôme donné en un point donné. Le conditionnement de l'évaluation a été étudié dans de nombreux articles pour des polynômes à coefficients complexes [206]. Cela signifie que l'on autorise des perturbations complexes sur les coefficients complexes du polynôme. Cependant, pour un polynôme à coefficients réels, il semble plus naturel de n'autoriser seulement que des perturbations réelles des coefficients réels. C'est par exemple le cas quand les coefficients du polynôme décrivent des valeurs physiques. Une autre motivation pour contraindre le conditionnement à décrire des perturbations réelles vient du calcul en précision finie. En effet, les erreurs d'arrondi d'un calcul sur des nombres réels en précision finie sont des nombres réels.

L'erreur inverse mesure la stabilité d'un algorithme numérique. Combinée avec le conditionnement, on obtient une borne sur l'erreur directe de la solution calculée par l'estimation empirique (1.1). Cette règle justifie l'étude simultanée de l'erreur inverse et du conditionnement pour le problème considéré ici. Nous considérons donc aussi l'étude de l'erreur inverse pour le problème de l'évaluation d'un polynôme donné en un point donné.

La principale contribution de ce chapitre est la définition et l'obtention d'une formule calculable pour le conditionnement et l'erreur inverse réelle pour l'évaluation polynomiale. Nous montrons ainsi qu'il y peu de différence entre le conditionnement réel et le conditionnement classique. Par contre, la différence peut être importante entre l'erreur inverse réelle et l'erreur inverse classique.

Le reste du chapitre est organisé de la manière suivante. Dans la section 2.1, nous considérons le conditionnement de l'évaluation d'un polynôme en un point donné. Nous considérons d'abord le conditionnement de l'évaluation d'un polynôme à coefficients complexes en un point complexe en autorisant des perturbations complexes sur les coefficients du polynôme. Nous nous intéressons ensuite au conditionnement de l'évaluation d'un polynôme à coefficients complexes en autorisant seulement des perturbations réelles sur les coefficients. Dans la section 2.2, nous considérons l'erreur inverse pour l'évaluation d'un polynôme en un point donné. Nous considérons tout d'abord le cas de polynômes à coefficients complexes et ensuite nous nous restreignons au cas des polynômes à coefficients réels.

2.1 Conditionnement de l'évaluation polynomiale

Dans cette section, nous considérons le conditionnement pour le problème de l'évaluation polynomiale en un point donné. Dans la première sous-section, nous rappelons les résultat sur le conditionnement pour des perturbations complexes. Dans la deuxième sous-section, nous introduisons le conditionnement réel pour l'évaluation polynomiale. Nous établissons une formule calculable pour ce conditionnement. Enfin, dans la dernière sous-section, nous étudions le lien entre le conditionnement de l'évaluation et la notion de pseudozéros.

2.1.1 Conditionnement de polynômes à coefficients complexes

Pour $n \geqslant 1$, on définit par $\mathcal{P}_n(\mathbf{C})$ l'espace des polynômes à coefficients complexes de degré au plus n. Soit $p \in \mathcal{P}_n(\mathbf{C})$ un polynôme de degré n donné par

$$p(z) = \sum_{i=0}^{n} p_i z^i.$$

Nous représenterons souvent p par le vecteur de ces coefficients $(p_0, p_1, \ldots, p_n)^T$. Nous identifierons aussi la norme $\|\cdot\|$ sur $\mathcal{P}_n(\mathbf{C})$ à la norme 2 sur $\mathbf{C}^{n+1}$ du vecteur correspondant. Dans ce chapitre nous travaillerons uniquement avec la norme 2 notée $\|\cdot\|$. Cela signifie que

$$\|p\| = \left(\sum_{i=0}^{n} |p_i|^2 \right)^{1/2}. \tag{2.1}$$

La définition classique pour le conditionnement de l'évaluation polynomiale en un point donné est la suivante.

Définition 2.1. Soit $p \in \mathcal{P}_n(\mathbf{C})$ un polynôme de degré n et z_0 un nombre complexe donné. Le conditionnement de l'évaluation de p en z_0 est

$$K^{\mathbf{C}}(z_0, p) := \lim_{\varepsilon \to 0} \sup \left\{ \frac{|\delta p(z_0)|}{\|\delta p\|} : \delta p \in \mathcal{P}_n(\mathbf{C}),\ \|\delta p\| = \varepsilon \right\}.$$

Une expression calculable pour le conditionnement est donnée par le théorème suivant.

Théorème 2.1 (Zhang [206]). *Soit $p \in \mathcal{P}_n(\mathbf{C})$ un polynôme de degré n et z_0 un nombre complexe donné. Le conditionnement pour l'évaluation polynomiale de p en z_0 vérifie*

$$K^{\mathbf{C}}(z_0, p) = \|\underline{z_0}\|,$$

avec $\underline{z_0} = (1, z_0, \ldots, z_0^n)^T$.

Démonstration. De l'inégalité de Hölder, on obtient

$$|\delta p(z_0)| = \left| \sum_{i=0}^{n} \delta p_i z_0^i \right| \leqslant \|\delta p\| \|\underline{z_0}\|.$$

On en déduit que $K^{\mathbf{C}}(z_0, p) \leqslant \|\underline{z_0}\|$. Il nous reste à prouver que la borne $\|\underline{z_0}\|$ peut être atteinte par un bon choix de δp. Notons $z_0 = |z_0|e^{i\theta}$ et $\delta p = (|z_0|^k e^{-ik\theta})_{k=0:n}$. Nous avons alors $\delta p \underline{z_0}^T = \|\underline{z_0}\|$ ce qui termine la preuve. ■

2.1.2 Conditionnement de polynômes à coefficients réels

Dans cette sous-section, nous autorisons seulement des perturbations réelles sur les coefficients d'un polynôme réel.

Définition 2.2. Soit $p \in \mathcal{P}_n(\mathbf{R})$ un polynôme de degré n et z_0 un nombre complexe. Le conditionnement réel de l'évaluation de p en z_0 est défini par

$$K^{\mathbf{R}}(z_0, p) := \lim_{\varepsilon \to 0} \sup \left\{ \frac{|\delta p(z_0)|}{\|\delta p\|} : \delta p \in \mathcal{P}_n(\mathbf{R}),\ \|\delta p\| = \varepsilon \right\}.$$

Une expression calculable pour le conditionnement réel est donnée par le théorème suivant.

Théorème 2.2. *Soit $p \in \mathcal{P}_n(\mathbf{R})$ un polynôme de degré n et z_0 un nombre complexe. Le conditionnement réel de l'évaluation de p en z_0 est*

$$K^{\mathbf{R}}(z_0, p) = \frac{1}{\sqrt{2}} \sqrt{(\|u\|^2 + \|v\|^2 + ((\|u\|^2 - \|v\|^2)^2 - 4(u^T v)^2)^{1/2})},$$

où u et v sont respectivement les parties réelles et imaginaires de $\underline{z_0}$.

Démonstration. D'après la définition 2.2, on a

$$K^{\mathbf{R}}(z_0, p) = \sup \left\{ |\delta p^T \underline{z_0}| : \delta p \in \mathcal{P}_n(\mathbf{R}),\ \|\delta p\| = 1 \right\},$$

et par conséquent,

$$K^{\mathbf{R}}(z_0, p)^2 = \sup \left\{ |\delta p^T \underline{z_0}|^2 : \delta p \in \mathcal{P}_n(\mathbf{R}),\ \|\delta p\| = 1 \right\}.$$

On utilise ici une méthode issue de la thèse de Hough [100]. Écrivons $\underline{z_0} = u + iv$, où u et v sont respectivement les parties réelles et imaginaires de $\underline{z_0}$. Avec ces notations, on a

$$|\underline{z_0}^T \delta p|^2 = \delta p^T (uu^T + vv^T)\delta p.$$

En appliquant le théorème du quotient de Rayleigh (voir [99, p.176]) on trouve

$$\sup\left\{|\underline{z_0}^T \delta p|^2,\ \delta p \in \mathcal{P}_n(\mathbf{R}),\ \|\delta p\| = 1\right\} = \lambda_{\max}(uu^T + vv^T),$$

où $\lambda_{\max}(A)$ dénote la plus grande valeur propre de la matrice A. La matrice $uu^T + vv^T$ est de rang 2 et son image est l'ensemble $\mathrm{Vect}(u, v)$. Soit λ une valeur propre de $uu^T + vv^T$ et $au + bv$, $a, b \in \mathbf{R}$ un vecteur propre. Il s'en suit que

$$(uu^T + vv^T)(au + bv) = \lambda(au + bv).$$

On peut réécrire cela sous la forme

$$\begin{pmatrix} u^T u & u^T v \\ v^T u & v^T v \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \lambda \begin{pmatrix} a \\ b \end{pmatrix}.$$

Les valeurs propres de cette matrice sont les racines de son polynôme caractéristique qui est

$$P(X) = X^2 - (u^T u + v^T v)X + (u^T u)(v^T v) - (u^T v)^2.$$

Les deux racines de ce polynôme sont

$$\begin{cases} X_1 = \frac{1}{2}(\|u\|^2 + \|v\|^2 + ((\|u\|^2 - \|v\|^2)^2 - 4(u^T v)^2)^{1/2}), \\ X_2 = \frac{1}{2}(\|u\|^2 + \|v\|^2 - ((\|u\|^2 - \|v\|^2)^2 - 4(u^T v)^2)^{1/2}). \end{cases}$$

En conséquence

$$\lambda_{\max}(uu^T + vv^T) = \frac{1}{2}(\|u\|^2 + \|v\|^2 + ((\|u\|^2 - \|v\|^2)^2 - 4(u^T v)^2)^{1/2}),$$

et donc $K^{\mathbf{R}}(z_0, p)^2 = \lambda_{\max}(uu^T + vv^T)$. Par conséquent,

$$K^{\mathbf{R}}(z_0, p)^2 = \frac{1}{2}(\|u\|^2 + \|v\|^2 + ((\|u\|^2 - \|v\|^2)^2 - 4(u^T v)^2)^{1/2}),$$

où u et v sont respectivement la partie réelle et imaginaire de $\underline{z_0}$, d'où le résultat. ■

Soit $p \in \mathcal{P}_n(\mathbf{R})$ un polynôme à coefficients réels. Comme

$$\begin{aligned} K^{\mathbf{R}}(z_0, p) &= \frac{1}{\sqrt{2}}\sqrt{(\|u\|^2 + \|v\|^2 + ((\|u\|^2 - \|v\|^2)^2 - 4(u^T v)^2)^{1/2})}, \\ K^{\mathbf{C}}(z_0, p) &= \sqrt{\|u\|^2 + \|v\|^2}, \end{aligned}$$

on en déduit

$$K^{\mathbf{C}}(z_0, p)/\sqrt{2} \leqslant K^{\mathbf{R}}(z_0, p) \leqslant K^{\mathbf{C}}(z_0, p). \tag{2.2}$$

Pour $p(z) = z + 1$ et $z_0 = 1$, on a $K^{\mathbf{R}}(z_0, p) = K^{\mathbf{C}}(z_0, p) = \sqrt{2}$. Pour $p(z) = z^3 + z$ et $z_0 = i$, on a $K^{\mathbf{R}}(z_0, p) = \sqrt{2}$ et $K^{\mathbf{C}}(z_0, p) = 2$. Par conséquent la relation (2.2) est optimale.

Remarque. Si le polynôme p est à coefficients réels et si z_0 est aussi réel, alors on doit avoir $K^{\mathbf{C}}(z_0, p) = K^{\mathbf{R}}(z_0, p)$. Cela se vérifie en prenant $v = 0$ dans les formules précédentes.

2.1.3 Conditionnement et ensembles des pseudozéros

Étant donné un réel $\varepsilon > 0$, le ε-voisinage de p est l'ensemble des polynômes de $\mathcal{P}_n(\mathbf{C})$, suffisamment proches de p, c'est-à-dire,

$$N_\varepsilon(p) = \{\widehat{p} \in \mathcal{P}_n(\mathbf{C}) : \|p - \widehat{p}\| \leqslant \varepsilon\}\,.$$

L'ensemble des ε-pseudozéros de p est l'ensemble des racines de tous les polynômes contenus dans le ε-voisinage de p, c'est-à-dire,

$$Z_\varepsilon(p) = \{z \in \mathbf{C} : \widehat{p}(z) = 0 \text{ pour } \widehat{p} \in N_\varepsilon(p)\}\,.$$

Le théorème 2.3 ci-dessous fournit une formule calculable pour caractériser cet ensemble.

Théorème 2.3 (Trefethen et Toh [192]). *L'ensemble des ε-pseudozéros de p vérifie*

$$Z_\varepsilon(p) = \left\{z \in \mathbf{C} : g(z) := \frac{|p(z)|}{\|\underline{z}\|} \leqslant \varepsilon\right\},$$

où $\underline{z} = (1, z, \ldots, z^n)^T$.

Zhang a remarqué que bien que le problème de l'évaluation polynomiale et la recherche de racines soient deux problèmes bien distincts, il y a une relation importante entre leur conditionnement. En effet, on définit le conditionnement relatif de l'évaluation de p en z_0 par

$$K^{\mathbf{C}}_{\mathrm{rel}}(z_0, p) := \lim_{\varepsilon \to 0} \sup \left\{ \frac{|\delta p(z_0)|/|p(z_0)|}{\|\delta p\|/\|p\|} : \delta p \in \mathcal{P}_n(\mathbf{R}),\ \|\delta p\| = \varepsilon \right\}.$$

On déduit immédiatement du théorème 2.1 que

$$K^{\mathbf{C}}_{\mathrm{rel}}(z_0, p) = \frac{\|p\|\|\underline{z_0}\|}{|p(z_0)|}.$$

Du théorème 2.3, on déduit alors le corollaire suivant.

Corollaire 2.1 (Zhang [206]). *Soit $p \in \mathcal{P}_n(\mathbf{C})$ un polynôme et $\varepsilon > 0$. On a*

$$Z_{\varepsilon\|p\|}(p) = \left\{z \in \mathbf{C} : \frac{1}{K^{\mathbf{C}}_{\mathrm{rel}}(z, p)} \leqslant \varepsilon\right\}.$$

2.2 Erreur inverse pour l'évaluation polynomiale

Dans cette section, nous considérons l'erreur inverse pour le problème de l'évaluation polynomiale. Comme dans les sections précédentes, nous regardons d'abord le problème avec des polynômes à coefficients complexes et nous autorisons des perturbations complexes de ces coefficients. Nous nous restreignons ensuite aux polynômes à coefficients réels et nous n'autorisons alors que des perturbations réelles de ces coefficients.

2.2.1 Erreur inverse de polynômes à coefficients complexes

Définition 2.3. Soient $p \in \mathcal{P}_n(\mathbf{C})$ un polynôme de degré n et $b \in \mathbf{C}$ une approximation de $p(x)$, $x \in \mathbf{C}$. L'erreur inverse de l'évaluation de p en x est définie par

$$\gamma_{p,x}^{\mathbf{C}}(b) := \min\{\|\delta p\| : \delta p \in \mathcal{P}_n(\mathbf{C}),\ (p+\delta p)(x) = b\}.$$

Lemme 2.1. *La solution du problème de minimisation*

$$\begin{cases} \min \|p\|^2, \\ \textit{tel que} \quad p(x) - b = 0, \end{cases}$$

où p est un polynôme de $\mathcal{P}_n(\mathbf{C})$ et où x et b sont deux nombres complexes fixés, est

$$\widehat{p}(z) = \frac{1}{\sum_{j=0}^n |x|^{2j}} \sum_{j=0}^n b\overline{x}^j z^j = \frac{1}{\|\underline{x}\|^2} \sum_{j=0}^n b\overline{x}^j z^j.$$

De plus, on a

$$\|\widehat{p}\| = \frac{|b|}{\sqrt{\sum_{j=0}^n |x|^{2j}}} = \frac{|b|}{\|\underline{x}\|^2}.$$

Démonstration. C'est un problème de minimisation strictement convexe si bien que la solution existe et est unique. Posons $p = \sum_{j=0}^n p_j z^j \in \mathcal{P}_n(\mathbf{C})$ avec $p_j = \alpha_j + i\beta_j$. Notons par $L(p, \lambda, \mu)$ le Lagrangien défini par

$$L(p, \lambda, \mu) = \|p\|^2 + 2\lambda \operatorname{Re}(p(x) - b) + 2\mu \operatorname{Im}(p(x) - b).$$

On remarque aisément que

$$\begin{cases} 2\operatorname{Re}(p(x) - b) = p(x) - b + \overline{p(x)} - \overline{b}, \\ 2\operatorname{Im}(p(x) - b) = -i(p(x) - b - \overline{p(x)} + \overline{b}). \end{cases}$$

La solution p satisfait au théorème des multiplicateurs de Lagrange ce qui signifie que

$$\frac{\partial L}{\partial \alpha_j} = 0 \quad \text{and} \quad \frac{\partial L}{\partial \beta_j} = 0 \quad \text{pour tout} \quad j = 0 : n.$$

Puisque nous avons les relations

$$\frac{\partial L}{\partial \alpha_j} = 2\alpha_j + (\lambda + i\mu)\overline{x}^j + (\lambda - i\mu)x^j = 0, \tag{2.3}$$

$$\frac{\partial L}{\partial \beta_j} = 2\beta_j + (-i\lambda + \mu)\overline{x}^j + (i\lambda + \mu)x^j = 0, \tag{2.4}$$

on obtient en faisant (2.3) + i(2.4),

$$p_j = -(\lambda + i\mu)\overline{x}^j, \quad \forall j = 0 : n.$$

De plus, en faisant (2.3)$x^j + i$(2.4)ix^j et en sommant sur j, on obtient

$$\lambda + i\mu = -\frac{p(x)}{\sum_{j=0}^{n} |x|^{2j}}.$$

Ainsi nous avons

$$p_j = \frac{b\overline{x}^j}{\sum_{k=0}^{n} |x|^{2k}} \quad \forall j = 0 : n.$$

■

Grâce au lemme précédent, on peut donner une expression explicite pour l'erreur inverse.

Théorème 2.4. *Soient $p \in \mathcal{P}_n(\mathbf{C})$ un polynôme de degré n et $b \in \mathbf{C}$ une approximation de $p(x)$, $x \in \mathbf{C}$. L'erreur inverse satisfait*

$$\gamma_{p,x}^{\mathbf{C}}(b) = \frac{|b - p(x)|}{\sqrt{\sum_{j=0}^{n} |x|^{2j}}} = \frac{|b - p(x)|}{\|\underline{x}\|}.$$

Démonstration. L'erreur inverse est la solution du problème de minimisation suivant

$$\begin{cases} \min \|\delta p\|^2, \\ \text{tel que} \quad \delta p(x) - (b - p(x)) = 0. \end{cases}$$

Le résultat est alors une conséquence directe du lemme 2.1. ■

Remarque. En fait, nous pouvons déduire ce résultat du calcul de l'erreur inverse sur les racines de polynôme.

Nous verrons au chapitre 10 qu'étant donné $p \in \mathcal{P}_n(\mathbf{C})$ un polynôme de degré n et z une racine approchée de p, on définie l'erreur inverse en z par

$$\eta_p^{\mathbf{C}}(z) := \min\{\|\delta p\| : \delta p \in \mathcal{P}_n(\mathbf{C}),\ (p + \delta p)(z) = 0\}.$$

On montrera aussi que $\eta_p^{\mathbf{C}}(z) = \frac{|p(z)|}{\|\underline{z}\|}$ où $\underline{z} := (1, z, z^2, \ldots, z^n)^T$. On remarque facilement que

$$\gamma_{p,x}^{\mathbf{C}}(b) = \min\{\|\delta p\| : \delta p \in \mathcal{P}_n(\mathbf{C}),\ ((p - b) + \delta p)(x) = 0\}.$$

On en déduit alors que $\gamma_{p,x}^{\mathbf{C}}(b) = \eta_{p-b}^{\mathbf{C}}(x)$ pour le polynôme $p - b$.

2.2.2 Erreur inverse de polynômes à coefficients réels

On s'intéresse maintenant au cas où le polynôme est à coefficients réels et où l'on n'autorise que des perturbations réelles de ses coefficients.

Définition 2.4. Soient $p \in \mathcal{P}_n(\mathbf{R})$ un polynôme de degré n et $b \in \mathbf{C}$ une approximation de $p(x)$, $x \in \mathbf{C}$. L'erreur inverse réelle de l'évaluation de p en x est définie par

$$\gamma_{p,x}^{\mathbf{R}}(b) := \min\{\|\delta p\| : \delta p \in \mathcal{P}_n(\mathbf{R}),\ (p + \delta p)(x) = b\}.$$

On peut alors donner une expression explicite pour cette erreur inverse.

Théorème 2.5. *Soient $p \in \mathcal{P}_n(\mathbf{R})$ un polynôme de degré n et $b \in \mathbf{C}$ une approximation de $p(x)$, $x \in \mathbf{C}$. L'erreur inverse réelle en x est donnée par*

$$\gamma_{p,x}^{\mathbf{R}}(b) = \frac{1}{d(G_R(x,b), \mathbf{R}G_I(x,b))},$$

où $G_R(x,b)$, $G_I(x,b)$ sont respectivement les parties réelles et imaginaires de

$$G(x,b) = \frac{1}{p(x)-b}(1, x, \ldots, x^n)^T.$$

Démonstration. Soient $p \in \mathcal{P}_n(\mathbf{R})$ un polynôme de degré n et z une racine approchée de p. L'erreur inverse réelle en z est

$$\eta_p^{\mathbf{R}}(z) := \min\{\|\delta p\| : \delta p \in \mathcal{P}_n(\mathbf{R}),\ (p+\delta p)(z) = 0\}.$$

Étant donnés $x, y \in \mathbf{R}^{n+1}$, on définit par

$$d(x, \mathbf{R}y) = \inf_{\alpha \in \mathbf{R}} \|x - \alpha y\|,$$

la distance du point $x \in \mathbf{R}^{n+1}$ à la droite $\mathbf{R}y = \{\alpha y, \alpha \in \mathbf{R}\}$. On montrera au chapitre 10 que l'erreur inverse réelle en z est donnée par $\eta_p^{\mathbf{R}}(z) = \frac{1}{d(G_R(z), \mathbf{R}G_I(z))}$, avec $G(z) = \frac{1}{p(z)}(1, z, \ldots, z^n)^T$ et G_R, G_I sont respectivement les parties réelles et imaginaires de $G(z)$. Il est clair alors que nous avons

$$\gamma_{p,x}^{\mathbf{R}}(b) = \min\{\|\delta p\| : \delta p \in \mathcal{P}_n(\mathbf{R}),\ ((p-b)+\delta p)(x) = 0\}.$$

Cela signifie que $\gamma_{p,x}^{\mathbf{R}}(b) = \eta_{p-b}^{\mathbf{R}}(x)$ pour le polynôme $p - b$. ■

Remarque. On peut se demander si $\gamma_{p,x}^{\mathbf{C}}(b) \approx \gamma_{p,x}^{\mathbf{R}}(b)$. En fait, $\gamma_{p,x}^{\mathbf{C}}(b)$ peut être très différent de $\gamma_{p,x}^{\mathbf{R}}(b)$. Si par exemple $p(x) = x + 1$ et $b = -1$, un calcul direct montre que $\gamma_{p,x}^{\mathbf{R}}(b) = 1$ alors que $\gamma_{p,x}^{\mathbf{C}}(b) = (1 + 1/|x|^2)^{-1/2}$ et donc $\gamma_{p,x}^{\mathbf{C}}(b) \to 0$ quand $x \to 0$.

2.3 Conclusion

Dans ce chapitre, nous nous sommes intéressés au problème classique de l'évaluation polynomiale sous l'aspect du conditionnement et de l'erreur inverse. Étant donné un polynôme à coefficients réels, nous avons défini un conditionnement et une erreur inverse réelle, c'est-à-dire que nous autorisons seulement des perturbations réelles sur ces coefficients. Nous avons donné une formule explicite pour ces expressions.

Nous avons surtout montré qu'il y a peu de différence entre le conditionnement classique et le conditionnement réel. Par contre, il peut y avoir une différence importante entre l'erreur inverse classique et l'erreur inverse réelle.

Ceci permet de justifier l'utilisation du conditionnement complexe même pour des polynômes à coefficients réels. Néanmoins, il faut être plus prudent lors de l'utilisation de l'erreur inverse complexe en lieu et place de l'erreur inverse réelle.

Chapitre 3

Évaluation précise de polynômes par la méthode de Horner

3.1 Introduction

Les polynômes apparaissent dans beaucoup de domaines du calcul scientifique et de l'ingénierie. Des applications en géométrie algorithmique et modélisation, mécanique, traitement du signal, ingénierie civile, robotique, simulation sont rapportées dans [68]. Développer des algorithmes rapides et fiables reste un des grands challenges actuels. Les méthodes de calcul numérique de racines se basent en général sur des méthodes itératives de type Newton. Ces méthodes nécessitent d'évaluer un polynôme et ses dérivées. Higham [87, chap. 5] consacre, par exemple, un chapitre entier aux polynômes et en particulier à l'évaluation polynomiale.

Comme nous l'avons vu dans le chapitre 2, le conditionnement de l'évaluation polynomiale peut être très élevé au voisinage d'une racine multiple. Par conséquent, l'évaluation par la méthode de Horner peut donner une erreur directe très grande. Néanmoins, la méthode de Horner est inverse-stable. L'idée de ce chapitre est donc de modifier l'algorithme de Horner afin de diminuer l'erreur directe.

Nous présentons un algorithme rapide et précis pour évaluer un polynôme univarié en précision finie. Par précis, nous entendons que la précision du résultat calculé est similaire à celle donnée par le schéma de Horner calculé avec une précision double de la précision courante. Par rapide, nous entendons que l'algorithme est plus rapide que les algorithmes existants donnant la même précision pour le résultat calculé. Nous montrons aussi comment calculer une borne d'erreur dynamique certifiée en précision finie pour vérifier la précision du résultat calculé. Tous ces algorithmes s'exécutent à une seule précision fixée (pas besoin de précision plus grande) et sont portables si l'on suppose que l'arithmétique flottante satisfait à la norme IEEE 754.

Puisque nous améliorons le schéma de Horner de façon similaire à la *sommation compensée* de Kahan [109], notre algorithme est présenté comme le *schéma de Horner compensé*. Les méthodes présentées dans ce chapitre sont fortement inspirées de l'article « Ac-

curate sum and dot product » d'Ogita, Rump et Oishi [153]. En particulier, les preuves sur les bornes d'erreurs utilisent les techniques introduites dans cet article.

3.1.1 Évaluation polynomiale numérique

Le schéma classique de Horner est un algorithme optimal d'un point de vue de la complexité pour évaluer un polynôme p dans la base canonique monomiale [197, p.371]. Il est implémenté dans bon nombre de logiciels et bibliothèques scientifiques, par exemple `SPOLY` et `DPOLY` dans IBM ESSL, `gs_poly_eval` dans GNU GSL, `polyval` dans MATLAB, etc. La faible erreur inverse introduite par le schéma de Horner dans l'évaluation en précision finie justifie son intérêt pratique en arithmétique flottante. On peut montrer [87, p.95] que l'évaluation en précision finie de $p(x)$ est l'évaluation exacte en x d'un polynôme obtenu en perturbant les coefficients de p avec une erreur relative de taille $2n\mathbf{u}$ où n est le degré du polynôme et $\mathbf{u}$ la précision du calcul.

La précision relative du résultat $\widehat{p}(x)$ calculé par le schéma de Horner vérifie l'estimation empirique (1.1) qui relie l'erreur directe au conditionnement et à l'erreur inverse. On rappelle que le conditionnement classique pour l'évaluation de $p(x) = \sum_{i=0}^{n} a_i x^i$ en x_0 est

$$\mathrm{cond}(p, x_0) = \frac{\sum_{i=0}^{n} |a_i||x_0|^i}{|\sum_{i=0}^{n} a_i x_0^i|} = \frac{\widetilde{p}(|x_0|)}{|p(x_0)|}.$$

On vérifie alors [87, p.95] que la précision relative du résultat calculé $\widehat{p}(x)$ est bornée par

$$\frac{|p(x) - \widehat{p}(x)|}{|p(x)|} \leqslant \alpha(n)\mathbf{u}\,\mathrm{cond}(p, x),$$

où $\alpha(n)$ est une fonction linéaire du degré n du polynôme (typiquement ici $\alpha(n) \approx 2n$). On retrouve un analogue de l'estimation empirique (1.1). Le problème est que le résultat calculé peut être extrêmement moins précis que la précision courante $\mathbf{u}$ si l'évaluation du polynôme est mal conditionnée. C'est par exemple le cas au voisinage d'une racine multiple où tous les chiffres du résultat calculé peuvent être faux.

Comment peut-on évaluer de manière plus précise un polynôme (arbitrairement) mal conditionné ? Avant de décrire les méthodes existantes, distinguons deux niveaux de mauvais conditionnement. Si l'on calcule avec une précision $\mathbf{u}$, l'évaluation de $p(x_0)$ est mal conditionnée si $1 \ll \mathrm{cond}(p, x_0) \leqslant \mathbf{u}^{-1}$. Nous dirons que l'évaluation est arbitrairement mal conditionnée si $\mathrm{cond}(p, x_0) \geqslant \mathbf{u}^{-1}$. Dans ce chapitre, nous considérons des polynômes à la fois mal conditionnés et arbitrairement mal conditionnés.

Si la précision $\mathbf{u}$ n'est pas suffisante pour garantir une précision donnée, on peut alors utiliser des bibliothèques multiprécision. Des expansions de longueur fixe telles que les « double-double » ou les « quad-double » [24, 9, 81] sont des solutions possibles pour simuler une précision double ou quadruple de la double précision IEEE 754. Par exemple, un quad-double est une somme non-evaluée de quatre nombres flottants double précision IEEE 754 ce qui donne au moins 212 bits de précision. C'est ce type d'expansions qui est utilisé dans les XBLAS [127]. Dans ce contexte, la façon la plus naturelle pour augmenter la précision d'un programme est de faire les calculs internes avec une précision étendue

et d'arrondir ensuite le résultat à la précision courante. Si le temps de calcul n'est pas une contrainte, des bibliothèques en précision arbitraire fournissent la solution (voir par exemple [2, 23, 165, 180]).

3.1.2 Le schéma de Horner compensé améliore l'estimation empirique classique

Nous présentons un algorithme précis et efficace pour l'évaluation de polynômes à une indéterminée. Nous nous intéressons au schéma de Horner et nous supposons que tous les calculs sont effectués à une précision fixée, en général la double précision IEEE 754. Par algorithme précis, nous entendons que la précision de la solution calculée est similaire à celle donnée par l'algorithme de Horner en précision supérieure (typiquement en utilisant des expansions de longueurs fixes). Là encore, la précision du résultat dépend toujours du conditionnement $\mathrm{cond}(p, x)$. Par efficace, nous entendons que l'algorithme soit au moins aussi rapide que la version calculée par des expansions de longueur fixe produisant un résultat de même précision. Notre algorithme (utilisant seulement la précision courante) est portable à condition que l'arithmétique réponde à la norme IEEE 754 ce qui est le cas de la plupart des arithmétiques utilisées de nos jours.

L'algorithme de Horner compensé que nous proposons ne requièrt ni branchement ni accès à la mantisse et utilise seulement la précision courante. Nous montrons que le résultat calculé `res` par le schéma de Horner compensé est aussi précis que s'il avait été calculé en précision doublée. Cela signifie que le résultat `res` satisfait une estimation empirique compensée,

$$\frac{|\mathtt{res} - p(x)|}{|p(x)|} \leqslant \mathbf{u} + \beta(n)\mathbf{u}^2\,\mathrm{cond}(p, x). \tag{3.1}$$

La fonction β est ici une fonction linéaire de n, le degré du polynôme et $\mathbf{u}$ est la précision du calcul. Le second terme de la partie droite de (3.1) reflète la précision obtenue avec un algorithme inverse-stable exécuté en précision double $\mathbf{u}^2$. Le premier terme vient lui de l'arrondi final à la précision courante $\mathbf{u}$.

Nous donnons aussi une borne d'erreur dynamique pour la précision du schéma de Horner compensé. On prouve que cette borne est valide et qu'elle est calculable en précision finie. Le schéma de Horner compensé et cette borne peuvent alors remplacer utilement le couple classique schéma de Horner/borne d'erreur dynamique quand une précision plus grande est nécessaire (voir Higham [87, chap. 5]). C'est par exemple le cas pour obtenir un critère d'arrêt pour la méthode de Newton au voisinage de racines multiples.

3.1.3 Utilisation de transformation exacte pour augmenter la précision

L'algorithme de Horner compensé permet de réduire les erreurs d'arrondi générées par les calculs en précision finie dans le schéma de Horner. Le point clé pour l'augmentation de cette précision est ce qu'Ogita, Rump et Oishi [153] ont appelé des *transformations*

exactes (« error-free transformations »). Cela signifie que pour deux nombres flottants a et b et une opération arithmétique $\circ = \{+, -, \cdot\}$ et $\mathrm{fl}(x)$ l'évaluation en flottant du réel x, il existe un nombre flottant e, calculable en arithmétique flottante, tel que

$$a \circ b = \mathrm{fl}(a \circ b) + e.$$

D'autres transformations exactes existent pour l'inverse et le `FMA`. Nous détaillerons ci-après les algorithmes correspondants dus à Kahan [109], Møller [140], Knuth [116], Dekker [46], Boldo et Muller [15].

En fait, l'augmentation de la précision résulte de la correction de l'erreur d'arrondi globale $p(x) - \widehat{p}(x)$, où $\widehat{p}(x)$ est le résultat du schéma de Horner en précision courante. Une telle correction a été précédemment expérimentée par Pichat [162] pour la somme et aussi mentionné dans [163] pour le schéma de Horner.

En utilisant les résultats de Tienari [188] et Linnainmaa [129, 131] sur les erreurs de linéarisation, Langlois a développé une méthode et un logiciel qui calculent une correction pour les erreurs d'arrondi au premier ordre [120, 123]. Il a aussi montré que le schéma de Horner peut être corrigé en calculant (exactement) le terme du premier ordre de $p(x) - \widehat{p}(x)$. Le calcul de la correction générique dans [120] se fait en utilisant les transformations exactes et la différentiation automatique.

Nous avons déjà mentionné que notre algorithme est au moins aussi rapide que si on exécutait l'algorithme classique de Horner en utilisant des expansions de longueur fixe tout en partageant la même précision pour le résultat calculé. L'efficacité pratique de l'utilisation des transformations exactes a été mise en évidence dans [153]. D'un point de vue théorique, le calcul dans les étapes de l'algorithme compensé est très similaire au calcul avec des « double-double » mais sans faire les rénormalisations nécessaires pour obtenir les non-chevauchements dans la représentation des « double-double ». En terme de performance, nos simulations montrent que le schéma de Horner compensé est plus de deux fois plus rapide que le schéma de Horner avec les « double-double ».

3.1.4 Organisation du chapitre

Le chapitre est organisé de la manière suivante. Dans la section 3.2, nous rappelons le modèle standard de l'arithmétique flottante IEEE 754 ainsi que quelques notations classiques. Dans la section 3.3, nous rappelons les transformations exactes pour la somme et le produit de deux nombres flottants. Dans la section 3.4, nous rappelons la méthode de Horner pour l'évaluation polynomiale ainsi que l'analyse d'erreur associée. Nous proposons ensuite une méthode de Horner compensée qui permet d'évaluer un polynôme avec une précision identique à celle obtenue si les calculs étaient effectués en précision double. Dans la section 3.5, nous présentons des résultats numériques montrant l'efficacité et la précision de l'algorithme proposé.

3.2 Modèle standard et notations

Dans ce chapitre, nous supposons que l'arithmétique flottante utilisée satisfait à la norme IEEE 754 (voir [103] et le chapitre 1). L'ensemble des nombres flottants est noté

$\mathbf{F}$, l'unité d'arrondi $\mathbf{u}$ et l'unité d'underflow $\underline{\mathbf{u}}$. On suppose que l'on n'a pas d'overflow dans les calculs. Pour la double précision IEEE 754, on a $\mathbf{u} = 2^{-53}$ and $\underline{\mathbf{u}} = 2^{-1074}$.

Nous noterons par $\mathrm{fl}(\cdot)$ le résultat d'un calcul où toutes les opérations effectuées à l'intérieur de la parenthèse sont effectuées dans la précision courante. Nous rappelons que, dans la norme IEEE 754, les opérations satisfont

$$\mathrm{fl}(a \circ b) = (a \circ b)(1+\varepsilon_1) = (a \circ b)/(1+\varepsilon_2) \text{ pour } \circ = \{+,-\} \text{ et } |\varepsilon_\nu| \leqslant \mathbf{u}$$

et

$$\mathrm{fl}(a \circ b) = (a \circ b)(1+\varepsilon_1) + \eta_1 = (a \circ b)/(1+\varepsilon_2) + \eta_2 \\ \text{pour } \circ = \{\cdot, /\} \text{ et } |\varepsilon_\nu| \leqslant \mathbf{u}, |\eta_\nu| \leqslant \underline{\mathbf{u}}.$$

L'addition et la soustraction sont exactes en cas d'underflow [79] et $\varepsilon_1\eta_1 = \varepsilon_2\eta_2 = 0$ pour la multiplication et la division [87, p.56]. Cela implique que

$$|a \circ b - \mathrm{fl}(a \circ b)| \leqslant \mathbf{u}|a \circ b| \text{ et } |a \circ b - \mathrm{fl}(a \circ b)| \leqslant \mathbf{u}|\,\mathrm{fl}(a \circ b)| \text{ pour } \circ = \{+,-\} \tag{3.2}$$

et

$$|a \circ b - \mathrm{fl}(a \circ b)| \leqslant \mathbf{u}|a \circ b| + \underline{\mathbf{u}} \text{ et} \\ |a \circ b - \mathrm{fl}(a \circ b)| \leqslant \mathbf{u}|\,\mathrm{fl}(a \circ b)| + \underline{\mathbf{u}} \text{ pour } \circ = \{\cdot, /\}. \tag{3.3}$$

On remarque que $a \circ b \in \mathbf{R}$ et $\mathrm{fl}(a \circ b) \in \mathbf{F}$ mais qu'en général on n'a pas $a \circ b \in \mathbf{F}$. On peut montrer que pour les opérations élémentaires $+, -, \cdot$, l'erreur d'approximation d'une opération flottante est encore un nombre flottant (voir par exemple [46]) :

$$\begin{array}{lcll} x = \mathrm{fl}(a \pm b) & \Rightarrow & a \pm b = x + y & \text{avec } y \in \mathbf{F}, \\ x = \mathrm{fl}(a \cdot b) & \Rightarrow & a \cdot b = x + y & \text{avec } y \in \mathbf{F}, \end{array} \tag{3.4}$$

si on suppose qu'il n'y a pas d'underflow pour la multiplication. Ce sont des transformations exactes du couple (a, b) en un couple (x, y).

Une méthode efficace pour prendre en compte les termes en puissance de $1+\varepsilon$ est décrite dans [87, p.67]. On notera $\langle k \rangle$ le compteur d'erreur défini par

$$\langle k \rangle = \prod_{i=1}^{k} (1+\varepsilon_i)^{\rho_i} \quad \text{avec} \quad \rho_i = \pm 1 \quad \text{et} \quad |\varepsilon_i| \leqslant \mathbf{u}.$$

Une règle utile pour ce compteur est $\langle j \rangle\langle k \rangle = \langle j \rangle / \langle k \rangle = \langle j + k \rangle$. Une autre quantité utile lors de majorations pour obtenir des bornes d'erreur est

$$\gamma_n := \frac{n\mathbf{u}}{1 - n\mathbf{u}} \quad \text{pour } n \in \mathbf{N}.$$

Les quantités $\langle n \rangle$ et γ_n sont liées par la relation

$$\langle n \rangle = 1 + \theta_n \text{ avec } |\theta_n| \leqslant \gamma_n.$$

En utilisant la notation γ_n, nous supposons toujours que $n\mathbf{u} < 1$. Pour une utilisation ultérieure, nous rappelons que nous avons les inégalités suivantes : $\gamma_n \leqslant \gamma_{n+1}$, $n\mathbf{u} \leqslant \gamma_n$ et

$$(1+\mathbf{u})\gamma_n = (1+\mathbf{u})\frac{n\mathbf{u}}{1-n\mathbf{u}} \leqslant \frac{n(\mathbf{u}+\mathbf{u}^2)}{1-(n+1)\mathbf{u}} \leqslant \frac{(n+1)\mathbf{u}}{1-(n+1)\mathbf{u}} = \gamma_{n+1}.$$

3.3 Transformations exactes

Dans cette section, nous montrons que l'erreur d'arrondi commise lors d'une opération d'addition ou de multiplication peut se calculer en arithmétique flottante.

Pour l'addition, nous utilisons l'algorithme suivant dû à Knuth [116, Thm B. p.236].

Algorithme 3.1. Transformation exacte de la somme de deux nombres flottants.

function $[x, y] = \texttt{TwoSum}(a, b)$
$x = \mathrm{fl}(a + b)$
$z = \mathrm{fl}(x - a)$
$y = \mathrm{fl}((a - (x - z)) + (b - z))$

Si l'on sait à l'avance que $|a| \geqslant |b|$, on peut utiliser l'algorithme suivant dû à Dekker [46].

Algorithme 3.2. Transformation exacte de la somme de deux nombres flottants.

function $[x, y] = \texttt{FastTwoSum}(a, b)$
$x = \mathrm{fl}(a + b)$
$y = \mathrm{fl}((a - x) + b)$

Pour la transformation exacte du produit, nous devons d'abord découper les flottants en entrée en deux parties. Soit p la longueur de la mantisse de la précision courante et soit $s = \lceil p/2 \rceil$. Par exemple, si la précision courante est la double précision IEEE 754, on a $p = 53$ et $s = 27$. L'algorithme suivant du à Dekker [46] découpe le flottant $a \in \mathbf{F}$ en deux flottants x, y. De plus, on a la relation

$$a = x + y \text{ et } x, y \text{ ne se chevauchent pas et } |y| \leqslant |x|.$$

Algorithme 3.3. Séparation exacte d'un nombre flottant en deux parties.

function $[x, y] = \texttt{Split}(a)$
$\texttt{factor} = 2^s + 1$
$c = \mathrm{fl}(\texttt{factor} \cdot a)$
$x = \mathrm{fl}(c - (c - a))$
$y = \mathrm{fl}(a - x)$

On peut maintenant formuler l'algorithme exact pour la multiplication attribué à Veltkamp (voir [46]).

Algorithme 3.4. Transformation exacte du produit de deux nombres flottants.

function $[x, y] = \texttt{TwoProduct}(a, b)$
$x = \mathrm{fl}(a \cdot b)$
$[a_1, a_2] = \texttt{Split}(a)$
$[b_1, b_2] = \texttt{Split}(b)$
$y = \mathrm{fl}(a_2 \cdot b_2 - (((x - a_1 \cdot b_1) - a_2 \cdot b_1) - a_1 \cdot b_2))$

Nous pouvons résumer les propriétés des algorithmes `TwoSum` et `TwoProduct` avec la proposition suivante.

Proposition 3.1. *Soit $a, b \in \mathbf{F}$ et $x, y \in \mathbf{F}$ vérifiant $[x, y] = \texttt{TwoSum}(a, b)$ (algorithme 3.1). Alors, même en présence d'un underflow,*

$$a + b = x + y, \quad x = \mathrm{fl}(a + b), \quad |y| \leqslant \mathbf{u}|x|, \quad |y| \leqslant \mathbf{u}|a + b|. \tag{3.5}$$

Soit $a, b \in \mathbf{F}$ et $x, y \in \mathbf{F}$ vérifiant $[x, y] = \texttt{TwoProduct}(a, b)$ (algorithme 3.4). Alors, en absence d'underflow,

$$a \cdot b = x + y, \quad x = \mathrm{fl}(a \cdot b), \quad |y| \leqslant \mathbf{u}|x|, \quad |y| \leqslant \mathbf{u}|a \cdot b|, \tag{3.6}$$

et, en présence d'un underflow,

$$a \cdot b = x + y + 5\eta, \quad x = \mathrm{fl}(a \cdot b), \quad |y| \leqslant \mathbf{u}|x| + 5\eta, \quad |y| \leqslant \mathbf{u}|a \cdot b| + 5\eta \text{ avec } |\eta| \leqslant \underline{\mathbf{u}}. \tag{3.7}$$

L'algorithme `TwoProduct` peut être réécrit de manière très simple si un Fused-Multiply-and-Add (`FMA`) est disponible sur l'architecture utilisée. C'est le cas par exemple avec l'Intel Itanium, l'IBM RS/6000 ou l'IBM PowerPC. Cela signifie que pour $a, b, c \in \mathbf{F}$, le résultat $\texttt{FMA}(a, b, c)$ est le nombre flottant le plus proche de la somme exacte $a \cdot b + c \in \mathbf{R}$ (voir [44, 45]). L'algorithme 3.4 peut alors être remplacé par l'algorithme suivant.

Algorithme 3.5. Transformation exacte d'un produit en utilisant un Fused-Multiply-and-Add.

function $[x, y] = \texttt{TwoProductFMA}(a, b)$
  $x = a \cdot b$
  $y = \texttt{FMA}(a, b, -x)$

Comme nous avons vu que le `FMA` pouvait être très utile, nous pouvons aussi penser à un opérateur qui calcule l'arrondi au plus près de la somme de trois nombres flottants. Supposons donc que l'on dispose de l'opérateur `ADD3` tel que pour $a, b, c \in \mathbf{F}$, $\texttt{ADD3}(a, b, c)$ soit le flottant le plus proche de $a + b + c \in \mathbf{R}$. On peut alors remplacer l'algorithme 3.1 par le suivant.

Algorithme 3.6. Transformation exacte d'une somme en utilisant `ADD3`.

function $[x, y] = \texttt{TwoSumADD3}(a, b)$
  $x = a + b$
  $y = \texttt{ADD3}(a, b, -x)$

Une transformation exacte pour le `FMA` a été récemment obtenue par Boldo et Muller [15]. L'erreur ne peut plus être représentée par un seul nombre flottant mais par la somme de deux nombres flottants.

Algorithme 3.7. Transformation exacte pour le Fused-Multiply-and-Add.

function $[x, y, z] = \texttt{ErrFMA}(a, b, c)$
$x = \mathrm{fl}(a \cdot b + c)$
$[u_1, u_2] = \texttt{TwoProduct}(a, b)$
$[\alpha_1, \alpha_2] = \texttt{TwoSum}(c, u_2)$
$[\beta_1, \beta_2] = \texttt{TwoSum}(u_1, \alpha_1)$
$\gamma = \mathrm{fl}((\beta_1 - x) + \beta_2)$
$[y, z] = \texttt{TwoSum}(\gamma, \alpha_2)$

3.4 Méthode de Horner compensée

Dans cette section, nous rappelons la méthode de Horner pour l'évaluation polynomiale ainsi que l'analyse d'erreur associée. Nous proposons ensuite une méthode de Horner compensée qui permet d'évaluer un polynôme avec une précision identique à celle obtenue si les calculs étaient effectués en précision double.

3.4.1 Méthode de Horner classique

L'algorithme classique pour évaluer un polynôme

$$p(x) = \sum_{i=0}^{n} a_i x^i$$

est l'algorithme de Horner rappelé ci-dessous.

Algorithme 3.8. Algorithme de Horner pour l'évaluation polynomiale

function $\texttt{res} = \texttt{Horner}(p, x)$
$s_n = a_n$
for $i = n - 1 : -1 : 0$
$s_i = \mathrm{fl}(s_{i+1} \cdot x + a_i)$
end
$\texttt{res} = s_0$

Une borne pour l'erreur directe de cet algorithme est (voir [87, p.95]) :

$$|p(x) - \texttt{res}| \leqslant \gamma_{2n} \sum_{i=0}^{n} |a_i||x|^i = \gamma_{2n} \widetilde{p}(|x|)$$

où $\widetilde{p}(x) = \sum_{i=0}^{n} |a_i| x^i$. Il est très intéressant d'exprimer et d'interpréter ce résultat en terme de conditionnement de l'évaluation polynomiale. On rappelle que ce conditionnement est défini par

$$\mathrm{cond}(p, x) := \lim_{\varepsilon \to 0} \sup \left\{ \frac{|(p + \Delta p)(x) - p(x)|}{\varepsilon |p(x)|} : \|\Delta p\| \leqslant \varepsilon \right\},$$

et vaut

$$\mathrm{cond}(p,x) = \frac{\sum_{i=0}^{n} |a_i||x|^i}{|p(x)|} = \frac{\widetilde{p}(|x|)}{|p(x)|}. \tag{3.8}$$

On en déduit que

$$\frac{|p(x) - \texttt{res}|}{|p(x)|} \leqslant \gamma_{2n}\,\mathrm{cond}(p,x).$$

On remarque donc que pour $\mathrm{cond}(p,x) > 1/\gamma_{2n}$, on ne peut garantir aucun chiffre du résultat calculé.

Si l'instruction `FMA` est disponible, alors on peut remplacer la ligne $s_i = \mathrm{fl}(s_{i+1}\cdot x + a_i)$ dans l'algorithme 3.8 par $s_i = \texttt{FMA}(s_{i+1}, x, a_i)$. Cela permet d'améliorer un peu la borne d'erreur car dans ce cas, le résultat calculé vérifie

$$\frac{|p(x) - \texttt{res}|}{|p(x)|} \leqslant \gamma_{n}\,\mathrm{cond}(p,x).$$

3.4.2 Méthode de Horner compensée

L'algorithme de Horner compensé que nous proposons ici consiste à calculer les erreurs d'arrondi commises dans l'algorithme d'Horner grâce aux fonctions `TwoSum` et `TwoProduct` et à les rajouter à la fin au résultat de l'algorithme de Horner classique. L'algorithme est décrit ci-dessous.

Algorithme 3.9. Algorithme de Horner compensé pour l'évaluation polynomiale

function `res` = `CompensatedHorner`(p,x)
$s_n = a_n$
$r_n = 0$
for $i = n-1 : -1 : 0$
  $[p_i, \pi_i] =$ `TwoProduct`(s_{i+1}, x)
  $[s_i, \sigma_i] =$ `TwoSum`(p_i, a_i)
  $r_i = \mathrm{fl}((r_{i+1}\cdot x + (\pi_i + \sigma_i))$
end
`res` $= s_0 + r_0$

Nous allons montrer que le résultat calculé par l'algorithme compensé 3.9 admet une borne d'erreur bien plus précise que celle de l'algorithme classique. En fait, l'algorithme 3.9 fournit un résultat comme si les calculs avaient été effectués en utilisant une précision double de celle de la précision courante. Pour prouver cette affirmation, nous avons besoin des résultats suivant.

Remarquons tout d'abord que d'après la proposition 3.1, on a $s_{i+1}\cdot x = p_i + \pi_i$ et $p_i + a_i = s_i + \sigma_i$. Par conséquent, on a

$$s_i = s_{i+1}\cdot x - \pi_i - \sigma_i \quad \text{pour } i = 0 : n-1.$$

On déduit alors par récurrence que

$$p(x) = s_0 + p_\pi(x) + p_\sigma(x), \tag{3.9}$$

avec

$$s_0 = \mathrm{fl}(p(x)), \quad p_\pi = \sum_{i=0}^{n-1} \pi_i x^i, \quad \text{et } p_\sigma = \sum_{i=0}^{n-1} \sigma_i x^i. \tag{3.10}$$

Dans la suite, nous noterons souvent $e(x) := p_\pi(x) + p_\sigma(x)$. Dans ce cas, on a $p(x) = \mathrm{fl}(p(x)) + e(x)$ et $\texttt{res} = \mathrm{fl}(p(x) + e(x))$.

Lemme 3.1. *Soit $p(x) = \sum_{i=0}^{n} a_i x^i$ un polynôme avec $a_i \in \mathbf{F}$, $0 \leqslant i \leqslant n$ et $x \in \mathbb{F}$. Soit p_π et p_σ définis dans (3.10). Alors on a, en absence d'underflow,*

$$\widetilde{p_\pi}(|x|) + \widetilde{p_\sigma}(|x|) \leqslant \gamma_{2n}\widetilde{p}(|x|),$$

avec $\widetilde{p}(x) = \sum_{i=0}^{n} |a_i| x^i$.

Démonstration. En utilisant la proposition 3.1 pour l'algorithme 3.9 on a pour $i = 1 : n$,

$$|p_{n-i}| = |\,\mathrm{fl}(s_{n-i+1} \cdot x)| \leqslant (1+\mathbf{u})|s_{n-i+1}||x|$$

et

$$|s_{n-i}| = |\,\mathrm{fl}(p_{n-i} + a_{n-i})| \leqslant (1+\mathbf{u})(|p_{n-i}| + |a_{n-i}|).$$

Montrons par récurrence sur $i = 1, \ldots, n$ que

$$|p_{n-i}| \leqslant (1+\gamma_{2i-1}) \sum_{j=1}^{i} |a_{n-i+j}||x^j| \tag{3.11}$$

et

$$|s_{n-i}| \leqslant (1+\gamma_{2i}) \sum_{j=0}^{i} |a_{n-i+j}||x^j|. \tag{3.12}$$

Pour $i = 1$, comme $s_n = a_n$, on a $|p_{n-1}| \leqslant (1+\mathbf{u})|a_n||x| \leqslant (1+\gamma_1)|a_n||x|$ et donc (3.11) est vraie. De la même façon, comme $|s_{n-1}| \leqslant (1+\mathbf{u})\left((1+\gamma_1)|a_n||x| + |a_{n-1}|\right) \leqslant (1+\gamma_2)\left(|a_n||x| + |a_{n-1}|\right)$, alors (3.12) est aussi vraie. Supposons maintenant que (3.11) et (3.12) sont vraies pour un entier i, $1 \leqslant i < n$. Alors on a

$$|p_{n-(i+1)}| \leqslant (1+\mathbf{u})|s_{n-i}||x|.$$

De l'hypothèse de récurrence, on déduit

$$\begin{aligned} |p_{n-(i+1)}| &\leqslant (1+\mathbf{u})(1+\gamma_{2i}) \sum_{j=0}^{i} |a_{n-i+j}||x^{j+1}| \\ &\leqslant (1+\gamma_{2(i+1)-1}) \sum_{j=1}^{i+1} |a_{n-(i+1)+j}||x^j|. \end{aligned}$$

Ainsi, on a

$$\begin{aligned}|s_{n-(i+1)}| &\leqslant (1+\mathbf{u})(|p_{n-(i+1)}| + |a_{n-(i+1)}|)\\ &\leqslant (1+\mathbf{u})(1+\gamma_{2(i+1)-1})\left[\sum_{j=1}^{i+1}|a_{n-(i+1)+j}||x^j| + |a_{n-(i+1)}|\right]\\ &\leqslant (1+\gamma_{2(i+1)})\sum_{j=0}^{i+1}|a_{n-(i+1)+j}||x^j|.\end{aligned}$$

Les relations (3.11) et (3.12) sont donc vraies par récurrence. Par conséquent, pour $i = 1:n$, on a

$$|p_{n-i}||x^{n-i}| \leqslant (1+\gamma_{2i-1})\widetilde{p}(x)$$

et

$$|s_{n-i}||x^{n-i}| \leqslant (1+\gamma_{2i})\widetilde{p}(x).$$

D'après la proposition 3.1, on a $|\pi_i| \leqslant \mathbf{u}|p_i|$ et $|\sigma_i| \leqslant \mathbf{u}|s_i|$ pour $i = 0:n-1$. Par conséquent,

$$(\widetilde{p_\pi} + \widetilde{p_\sigma})(|x|) = \sum_{i=0}^{n-1}(|\pi_i| + |\sigma_i|)|x^i| \leqslant \mathbf{u}\sum_{i=1}^{n}(|p_{n-i}| + |\sigma_{n-i}|)|x^{n-i}|,$$

et donc

$$(\widetilde{p_\pi} + \widetilde{p_\sigma})(|x|) \leqslant \mathbf{u}\sum_{i=1}^{n}(2+\gamma_{2i-1}+\gamma_{2i})\,\widetilde{p}(|x|) \leqslant 2n\mathbf{u}\,(1+\gamma_{2n})\,\widetilde{p}(|x|).$$

Puisque $2n\mathbf{u}(1+\gamma_{2n}) = \gamma_{2n}$, on obtient finalement $(\widetilde{p_\pi} + \widetilde{p_\sigma})(|x|) \leqslant \gamma_{2n}\widetilde{p}(|x|)$. ■

Lemme 3.2. *Soit $p(x) = \sum_{i=0}^{n} a_i x^i$ un polynôme avec $a_i \in \mathbf{F}$, $0 \leqslant i \leqslant n$, $q(x) = \sum_{i=0}^{n} b_i x^i$ un polynôme avec $b_i \in \mathbf{F}$, $0 \leqslant i \leqslant n$ et $x \in \mathbb{F}$. Alors l'évaluation flottante de $r(x) = p(x) + q(x)$ selon l'algorithme suivant*

$r_n = \mathrm{fl}(a_n + b_n)$
for $i = n-1:-1:0$
  $r_i = \mathrm{fl}(r_{i+1}\cdot x + (a_i + b_i))$
end
`res` $= r_0$

satisfait, en absence d'underflow,

$$|\texttt{res} - r(x)| \leqslant \gamma_{2n+1}\widetilde{r}(|x|).$$

Démonstration. En considérant l'algorithme de calcul, on a $r_n = \mathrm{fl}(a_n + b_n) = (a_n + b_n)\langle 1\rangle$ et pour $i = n-1:0$,

$$r_i = \mathrm{fl}(r_{i+1}\cdot x + (a_i + b_i)) = r_{i+1}x\langle 2\rangle + (a_i + b_i)\langle 2\rangle.$$

Par conséquent, on montre alors par récurrence que

$$r_0 = (a_n + b_n)x^n\langle 2n+1\rangle + \sum_{i=0}^{n-1}(a_i + b_i)x^i\langle 2(i+1)\rangle.$$

Par conséquent, avec les quantités $\theta_{2n+1}, \theta_{2n}, \cdots, \theta_1$, vérifiant $|\theta_i| \leqslant \gamma_i$, on a

$$r_0 = (a_n + b_n)x^n(1+\theta_{2n+1}) + \sum_{i=0}^{n-1}(a_i + b_i)x^i(1+\theta_{2(i+1)}).$$

Puisque $r_0 = \mathrm{fl}(p(x) + q(x))$, on obtient finalement

$$\left|\texttt{res} - \sum_{i=0}^{n}(a_i + b_i)x^i\right| \leqslant \gamma_{2n+1}\sum_{i=0}^{n}|a_i + b_i||x^i| \leqslant \gamma_{2n+1}(\widetilde{p} + \widetilde{q})(|x|),$$

ce qui est bien le résultat voulu. ■

Théorème 3.1. *Soit $p(x) = \sum_{i=0}^{n} a_i x^i$ un polynôme avec $a_i \in \mathbf{F}$, $0 \leqslant i \leqslant n$ et $x \in \mathbf{F}$. Soit* `res` *le résultat calculé par l'algorithme 3.9. Alors, en absence d'underflow,*

$$|\texttt{res} - p(x)| \leqslant \mathbf{u}|p(x)| + \gamma_{2n}^2\widetilde{p}(|x|). \tag{3.13}$$

Démonstration. En considérant l'algorithme 3.9, on a $p(x) = s_0 + e(x)$. On en déduit que

$$\begin{aligned}
|\texttt{res} - p(x)| &= |(1+\varepsilon)(s_0 + \mathrm{fl}(e(x))) - p(x)| \\
&= |(1+\varepsilon)(s_0 + \mathrm{fl}(e(x)) - p(x)) + \varepsilon p(x)| \\
&= |(1+\varepsilon)(s_0 + e(x) - p(x)) + (1+\varepsilon)(\mathrm{fl}(e(x)) - e(x)) + \varepsilon p(x)| \\
&\leqslant \mathbf{u}|p(x)| + (1+\mathbf{u})|\,\mathrm{fl}(e(x)) - e(x)|.
\end{aligned}$$

En appliquant le lemme 3.2, on obtient

$$|\,\mathrm{fl}(e(x)) - e(x)| \leqslant \gamma_{2n-1}\widetilde{e}(|x|) = \gamma_{2n-1}(\widetilde{p_\pi}(|x|) + \widetilde{p_\sigma}(|x|)).$$

De plus d'après le lemme 3.1, on a

$$\widetilde{p_\pi}(|x|) + \widetilde{p_\sigma}(|x|) \leqslant \gamma_{2n}\widetilde{p}(|x|).$$

Par conséquent, on en déduit

$$|\texttt{res} - p(x)| \leqslant \mathbf{u}|p(x)| + (1+\mathbf{u})\gamma_{2n-1}\gamma_{2n}\widetilde{p}(|x|).$$

Puisque $(1+\mathbf{u})\gamma_{2n-1} \leqslant \gamma_{2n}$, on obtient

$$|\texttt{res} - p(x)| \leqslant \mathbf{u}|p(x)| + \gamma_{2n}^2\widetilde{p}(|x|),$$

ce qui est le résultat souhaité. ■

Le résultat final « res » est la somme flottante de deux nombres flottantes (voir la dernière ligne de l'algorithme 3.9). Le résultat exacte « $p(x)$ » n'est pas en général un nombre flottant. C'est pourquoi nous ne pouvons pas obtenir une borne d'erreur (3.13) meilleure que $\mathbf{u}|p(x)|$. La borne d'erreur (3.13) nous dit que la qualité du résultat « res » est comme s'il avait été calculé dans une précision double de la précision courante et ensuite arrondi à la précision courante.

En combinant le résultat du théorème 3.1 avec l'expression du conditionnement (3.8), on obtient le résultat suivant.

Corollaire 3.1. *Soit $p(x) = \sum_{i=0}^{n} a_i x^i$ un polynôme avec $a_i \in \mathbf{F}$, $0 \leqslant i \leqslant n$ et $x \in \mathbf{F}$. Soit* res *le résultat calculé par l'algorithme 3.9. Alors, en absence d'underflow,*

$$\frac{|\mathtt{res} - p(x)|}{|p(x)|} \leqslant \mathbf{u} + \gamma_{2n}^2 \operatorname{cond}(p, x). \tag{3.14}$$

3.4.3 Borne calculable de l'erreur

La borne d'erreur (3.13) pour le résultat res de l'algorithme 3.9 n'est pas calculable puisqu'elle fait intervenir la valeur exacte $p(x)$. Dans la suite, nous montrons comment calculer une borne d'erreur valide en flottant avec arrondi au plus près. Nous verrons expérimentalement que cette borne est plus précise dans celle de (3.13).

En suivant la preuve du théorème 3.1, on a

$$\begin{aligned}
|\mathtt{res} - p(x)| &\leqslant |\operatorname{fl}(p(x) + e(x)) - p(x)| \\
&\leqslant |\operatorname{fl}(p(x) + e(x)) - (\operatorname{fl}(p(x)) + \operatorname{fl}(e(x))| + |\operatorname{fl}(p(x)) + \operatorname{fl}(e(x)) - p(x)| \\
&\leqslant \mathbf{u}|\mathtt{res}| + |\operatorname{fl}(e(x)) - e(x)| \\
&\leqslant \mathbf{u}|\mathtt{res}| + \gamma_{2n-1}\widetilde{e}(|x|) \\
&\leqslant \mathbf{u}|\mathtt{res}| + \gamma_{2n-1}(1+\mathbf{u})^{2n-1} \operatorname{fl}(\widetilde{e}(|x|)).
\end{aligned}$$

Pour $m\mathbf{u} < 1$ avec $m \in \mathbf{F}$, on a $\operatorname{fl}(m\mathbf{u}) = m\mathbf{u} \in \mathbf{F}$ et $\operatorname{fl}(1a - m\mathbf{u}) = 1 - m\mathbf{u} \in \mathbf{F}$. Par conséquent, seule la division peut causer une erreur d'arrondi dans le calcul de $\gamma_n = m\mathbf{u}/(1 - m\mathbf{u})$. On en déduit donc que

$$\gamma_n \leqslant (1-\mathbf{u})^{-1} \operatorname{fl}(m\mathbf{u}/(1-m\mathbf{u})).$$

Un calcul simple montre aussi que pour $m \in \mathbf{N}$, on a $\gamma_m \leqslant (1-\mathbf{u})\gamma_{m+1}$ et que

$$\widetilde{e}(|x|) \leqslant (1+\mathbf{u})^{2n-1} \operatorname{fl}(\widetilde{e}(|x|)).$$

On déduit de tout cela que

$$\begin{aligned}
|\mathtt{res} - p(x)| &\leqslant \operatorname{fl}(\mathbf{u}|\mathtt{res}|) + \gamma_{4n+2}(1-\mathbf{u})^4 \operatorname{fl}(\widetilde{e}(|x|)) \\
&\leqslant (1-\mathbf{u}) \operatorname{fl}(\mathbf{u}|\mathtt{res}|) + (1-\mathbf{u})^3 \operatorname{fl}(\gamma_{4n+2}) \operatorname{fl}(\widetilde{e}(|x|)) + \mathbf{u} \operatorname{fl}(\mathbf{u}|\mathtt{res}|) \\
&\leqslant (1-\mathbf{u}) \operatorname{fl}(\mathbf{u}|\mathtt{res}|) + (1-\mathbf{u})^2 \operatorname{fl}(\gamma_{4n+2}\widetilde{e}(|x|)) \\
&\quad + (1-\mathbf{u})^2 \operatorname{fl}(2\mathbf{u}^2|\mathtt{res}|) \\
&\leqslant (1-\mathbf{u}) \operatorname{fl}(\mathbf{u}|\mathtt{res}|) + (1-\mathbf{u}) \operatorname{fl}(\gamma_{4n+2}\widetilde{e}(|x|) + 2\mathbf{u}^2|\mathtt{res}|) \\
&\leqslant \operatorname{fl}(\mathbf{u}|\mathtt{res}| + (\gamma_{4n+2}\widetilde{e}(|x|) + 2\mathbf{u}^2|\mathtt{res}|)).
\end{aligned}$$

En résumé, on a le résultat suivant.

Proposition 3.2. *Soit* $p(x) = \sum_{i=0}^{n} a_i x^i$ *un polynôme avec* $a_i \in \mathbf{F}$, $0 \leqslant i \leqslant n$ *et* $x \in \mathbf{F}$. *Soit* `res` *le résultat calculé par l'algorithme 3.9. Alors, en absence d'underflow,*

$$|\mathtt{res} - p(x)| \leqslant \mathrm{fl}(\mathbf{u}|\mathtt{res}| + (\gamma_{4n+2}\,\mathrm{fl}((\widetilde{p_\pi} + \widetilde{p_\sigma})(|x|)) + 2\mathbf{u}^2|\mathtt{res}|)). \tag{3.15}$$

Si l'on note

$$\mathtt{err} = \mathbf{u}|\mathtt{res}| + (\gamma_{4n+2}\,\mathrm{fl}((\widetilde{p_\pi} + \widetilde{p_\sigma})(|x|)) + 2\mathbf{u}^2|\mathtt{res}|), \tag{3.16}$$

alors on a

$$\mathtt{res} - \mathtt{err} \leqslant p(x) \leqslant \mathtt{res} + \mathtt{err}.$$

3.5 Simulations numériques

Toutes les simulations sont effectuées en utilisant la double précision IEEE 754.

3.5.1 Le schéma de Horner avec les double-double

Nous allons comparer notre algorithme 3.9 d'Horner compensé avec l'implémentation classique du schéma de Horner en utilisant en interne le format double-double développé dans [9, 81, 127]. Dans notre cas, il est suffisant de savoir qu'un double-double a est un couple (a_h, a_l) de doubles vérifiant $a = a_h + a_l$ et $|a_l| \leqslant \mathbf{u}|a_h|$. Pour implémenter l'algorithme de Horner avec le format double-double, il suffit de savoir faire les deux opérations suivantes :

(i) le produit d'un double-double avec un double, et

(ii) l'addition d'un double avec un double-double.

Pour (i), nous utilisons l'algorithme 3.10. Pour (ii) nous utilisons l'algorithme 3.11. L'algorithme de Horner avec le format double-double est présenté dans l'algorithme 3.12 ci-dessous.

Algorithme 3.10. Produit d'un double-double (a_h, a_l) par un double b.

function $[c_h, c_l] = \mathtt{prod\_dd\_d}(a_h, a_l, b)$
$[s_h, s_l] = \mathtt{TwoProduct}(a_h, b)$
$[t_h, t_l] = \mathtt{FastTwoSum}(s_h, \mathrm{fl}(a_l \cdot b))$
$[c_h, c_l] = \mathtt{FastTwoSum}(t_h, \mathrm{fl}(t_l + s_l))$

Algorithme 3.11. Addition d'un double b avec un double-double (a_h, a_l).

function $[c_h, c_l] = \mathtt{add\_dd\_d}(a_h, a_l, b)$
$[t_h, t_l] = \mathtt{TwoSum}(a_h, b)$
$[c_h, c_l] = \mathtt{FastTwoSum}(t_h, \mathrm{fl}(t_l + a_l))$

Algorithme 3.12. Algorithme de Horner avec le format double-double.

TAB. 3.1 – Description des routines testées

routine	description de l'algorithme correspondant
`Horner`	double précision IEEE 754
`CompensatedHorner`	notre algorithme compensé 3.9
`DDHorner`	schéma de Horner avec le format double-double 3.12
`MPFRHorner`	schéma de Horner avec 106 bits de précision avec MPFR

function `res` = `DDHorner`(p, x)
$s_h = a_n$
$s_l = 0$
for $i = n-1 : -1 : 0$
  $[p_h, p_l]$ = `prod_dd_d`(s_h, s_l, x)
  $[s_h, s_l]$ = `add_dd_d`(p_h, p_l, a_i)
end
`res` $= s_h$

3.5.2 Précision de l'algorithme de Horner compensé

Nous avons testé notre algorithme avec les polynômes $p_n(x) = (x-1)^n$ sous leurs formes développées. La variable x est choisie proche de la racine 1 de p_n et avec beaucoup de chiffres significatifs afin que de nombreuses erreurs d'arrondi apparaissent dans les calculs. Nous faisons varier le degré n de 1 jusqu'à une valeur suffisante pour couvrir une large plage de conditionnements $\mathrm{cond}(p_n, x)$. Dans notre cas, on a

$$\mathrm{cond}(p_n, x) = \frac{\widetilde{p_n}(|x|)}{|p_n(x)|} = \left|\frac{1+|x|}{1-x}\right|^n,$$

et donc $\mathrm{cond}(p_n, x)$ augmente de façon exponentielle par rapport à n. Dans les simulations reportées dans la figure 3.1, $\mathrm{cond}(p_n, x)$ varie de 10^2 à 10^{40} (pour $x = \mathrm{fl}(1.333)$, ce qui correspond à une plage de degré variant de 3 à 42.

Nous avons comparé les algorithmes `Horner` et `CompensatedHorner` (voir la table 3.1). Pour chaque polynôme p_n, la valeur exacte $p_n(x)$ est approchée avec une grande précision grâce la Symbolic Toolbox de MATLAB. La figure 3.1 représente la précision relative $|y - p_n(x)|/|p_n(x)|$ de l'évaluation y calculée par ces deux algorithmes.

Nous observons que notre algorithme a bien le comportement attendu, c.-à-d. vérifie bien l'estimation empirique compensée. Tant que le conditionnement est inférieur à $\mathbf{u}^{-1} \approx 10^{16}$, le résultat est calculé une précision de l'ordre de la précison machine. Pour un conditionnement compris entre $\mathbf{u}^{-1}$ et $\mathbf{u}^{-2} \approx 10^{32}$, l'erreur relative se dégrade jusqu'à ne plus avoir de précision du tout. On remarque aussi que quand le conditionnement est supérieure à $\mathbf{u}^{-1}$, la borne d'erreur *a priori* (3.14) est toujours pessimiste d'un ordre de grandeur de 2 ou 3.

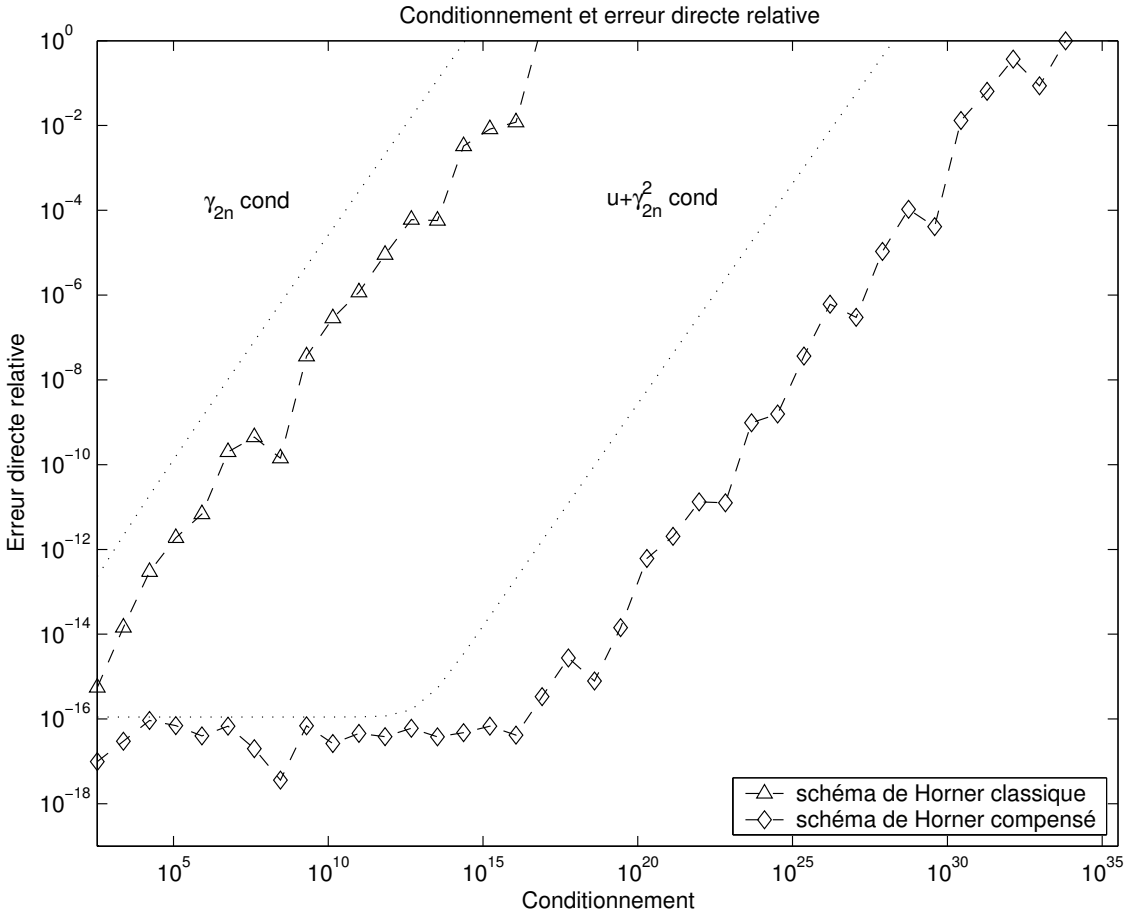


FIG. 3.1 – Précision relative du schéma de Horner en double précision IEEE 754 pour les algorithmes `Horner` et `CompensatedHorner`.

3.5.3 Précision de la borne d'erreur dynamique

Nous comparons ici la précision de notre borne d'erreur dynamique (3.15) avec la borne d'erreur *a priori* (3.14) et l'erreur directe réelle. Pour ce faire, nous évaluons le polynôme (sous sa forme développée) $p_5(x) = (1-x)^5$ en 1024 points autour de $x = 1$. Pour chaque valeur de x, on calcule `CompensatedHorner`(p_5, x), la borne d'erreur dynamique, et l'erreur inverse réelle. Les résultats sont rapportés dans la figure 3.2.

On observe que plus x est proche de la racine 1 (c.-à-d. plus le conditionnement est grand), moins la borne d'erreur *a priori* est précise. Notre borne d'erreur dynamique est, quant à elle, plus précise puisqu'elle prend en compte les erreurs d'arrondi qui ont eu lieu lors du calcul.

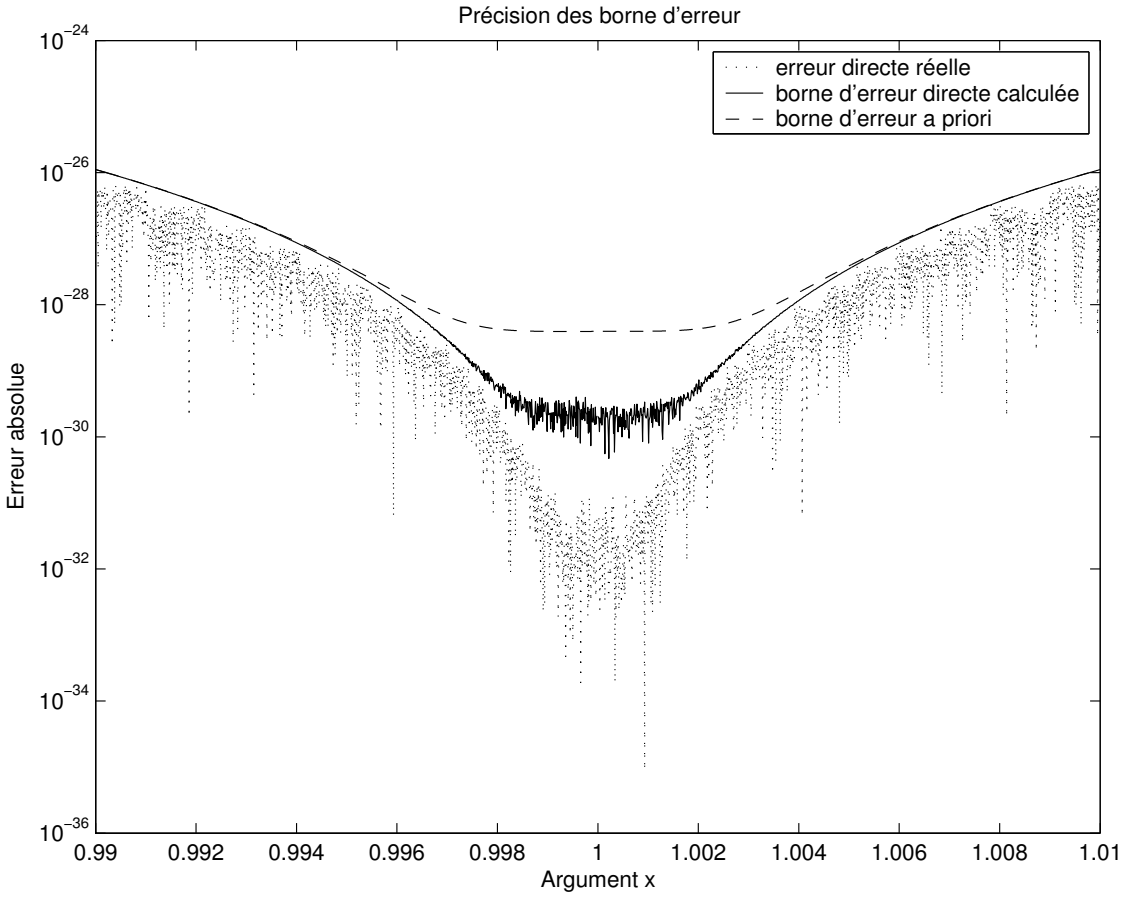


FIG. 3.2 – Comparaisons des bornes d'erreur *a priori* et dynamique avec l'erreur directe réelle.

3.5.4 Performance

Tous les algorithmes ont été implémentés en langage C. Les environnement utilisés pour les tests sont décrits dans la table 3.2. Nos mesures ont été effectuées sur des poly-

TAB. 3.2 – Environnement de simulations.

environnement	description
(I)	Intel Celeron, 2.4GHz, 1024kB L2 cache. GNU Compiler Collection 3.4.1
(II)	Intel Pentium, 3.0GHz, 1024kB L2 cache. GNU Compiler Collection 3.4.1

TAB. 3.3 – Performances en temps mesuré pour `CompensatedHorner`, `DDHorner` et `MPFRHorner`.

environnement	`CompensatedHorner`/`Horner`				`DDHorner`/`Horner`				`MPFRHorner`/`Horner`		
	min.	moyen	max.	theo.	min.	moyen	max.	theo.	min.	moyen	max.
(I)	1.4	3.1	3.4	13	2.3	8.4	9.4	17	22.4	97.5	124.4
(II)	1.5	2.9	3.2	13	2.3	8.4	9.4	17	18.1	83.7	96.8

nômes dont le degré varie de 5 à 500 par pas de 5. Nous avons choisi les coefficients et la variable à évaluer de manière aléatoire. Pour chaque degré, les routines sont testées avec le même polynôme et la même variable. Le minimum, la moyenne et le maximum des temps de calcul sont reportés dans la table 3.3. Pour `CompensatedHorner` et `DDHorner`, on a aussi reporté les ratios théoriques résultant du nombre de flops de chaque algorithme.

On remarque tout d'abord que le ratio effectivement mesuré pour `CompensatedHorner` ou `DDHorner` est plus faible que le ratio théorique. Ceci est assez étonnant puisque les algorithmes `CompensatedHorner` et `DDHorner` ne sont pas optimisés afin de les rendre portables. Nous pensons que cela vient du fait que l'algorithme de Horner classique fait seulement une opération avec chaque coefficient du polynôme alors que les autres algorithmes font plusieurs opérations avec chaque coefficient. La plupart des opérations sont effectuées au niveau des registres sans occasionner de coûteux transferts avec la mémoire.

Les résultats reportés dans la table 3.3 montrent que notre algorithme de Horner compensé `CompensatedHorner` est à peu près 3 fois plus lent que l'algorithme de Horner classique. L'algorithme `DDHorner` est quant à lui 8 fois plus lent que l'algorithme de Horner classique `Horner`. On peut donc dire que notre algorithme est plus de 2 fois plus rapide que l'algorithme de Horner avec les « double-doubles ». La comparaison avec MPFR n'est pas tout à fait honnête (un facteur de l'ordre de 80) puisque MPFR est faite pour gérer des nombres avec des mantisses extrêmement grandes.

3.6 Le cas de l'underflow

Remarquons tout d'abord que d'après la proposition 3.1, on a $s_{i+1} \cdot x = p_i + \pi_i + 5\eta_i$ avec $|\eta_i| \leqslant \underline{\mathbf{u}}$ et $p_i + a_i = s_i + \sigma_i$. Par conséquent, on a

$$s_i = s_{i+1} \cdot x - \pi_i - \sigma_i - 5\eta_i \quad \text{pour } i = 0 : n-1.$$

On déduit alors par récurrence que

$$p(x) = s_0 + p_\pi(x) + p_\sigma(x) + 5\sum_{i=0}^{n-1} \eta_i x^i, \tag{3.17}$$

avec

$$s_0 = \mathrm{fl}(p(x)), \quad p_\pi = \sum_{i=0}^{n-1} \pi_i x^i, \quad \text{et } p_\sigma = \sum_{i=0}^{n-1} \sigma_i x^i. \tag{3.18}$$

Dans la suite, nous noterons souvent $e(x) := p_\pi(x) + p_\sigma(x)$. Dans ce cas, on a $p(x) = \mathrm{fl}(p(x)) + e(x)$ et $\texttt{res} = \mathrm{fl}(p(x) + e(x))$.

Lemme 3.3. *Soit $p(x) = \sum_{i=0}^{n} a_i x^i$ un polynôme avec $a_i \in \mathbf{F}$, $0 \leqslant i \leqslant n$ et $x \in \mathbb{F}$. Soit p_π et p_σ définis dans (3.10). Alors on a, en présence d'underflow,*

$$\widetilde{p_\pi}(|x|) + \widetilde{p_\sigma}(|x|) \leqslant \gamma_{2n}\widetilde{p}(|x|) + (5 + \gamma_{2n})\underline{\mathbf{u}} \sum_{i=0}^{n-1} |x|^i.$$

Démonstration. En utilisant le modèle standard de l'arithmétique flottante avec underflow, on a pour $i = 1, \ldots, n$ (voir l'algorithme 3.9)

$$|p_{n-i}| = |\,\mathrm{fl}(s_{n-i+1} \cdot x)| \leqslant (1 + \mathbf{u})|s_{n-i+1}||x| + \underline{\mathbf{u}},$$

et

$$|s_{n-i}| = |\,\mathrm{fl}(p_{n-i} + a_{n-i})| \leqslant (1 + \mathbf{u})(|p_{n-i}| + |a_{n-i}|).$$

Montrons maintenant par récurrence que pour $i = 1, \ldots, n$, on a

$$|p_{n-i}| \leqslant (1 + \gamma_{2i-1}) \sum_{j=1}^{i} |a_{n-i+j}||x^j| + (1 + \gamma_{2i-2})\underline{\mathbf{u}} \sum_{j=0}^{i-1} |x^j|, \tag{3.19}$$

$$|s_{n-i}| \leqslant (1 + \gamma_{2i}) \sum_{j=0}^{i} |a_{n-i+j}||x^j| + (1 + \gamma_{2i-1})\underline{\mathbf{u}} \sum_{j=0}^{i-1} |x^j|. \tag{3.20}$$

Pour $i = 1$, comme $s_n = a_n$, on a $|p_{n-1}| \leqslant (1+\mathbf{u})|a_n||x| + \underline{\mathbf{u}} \leqslant (1+\gamma_1)|a_n||x| + \underline{\mathbf{u}}$ et donc la relation (3.19) est vraie. De la même façon, nous avons

$$|s_{n-1}| \leqslant (1 + \mathbf{u})\left((1 + \gamma_1)|a_n||x| + \underline{\mathbf{u}} + |a_{n-1}|\right) \leqslant (1 + \gamma_2)\left(|a_n||x| + |a_{n-1}|\right) + (1 + \gamma_1)\underline{\mathbf{u}},$$

et la relation (3.20) est aussi vraie. Supposons maintenant que (3.19) et (3.20) sont vraie pour un entier i tel que $1 \leqslant i < n$. Alors

$$|p_{n-(i+1)}| \leqslant (1 + \mathbf{u})|s_{n-i}||x| + \underline{\mathbf{u}}.$$

Par hypothèse de récurrence, on a

$$\begin{aligned} |p_{n-(i+1)}| &\leqslant (1 + \mathbf{u})(1 + \gamma_{2i}) \sum_{j=0}^{i} |a_{n-i+j}||x^{j+1}| + (1 + \mathbf{u})(1 + \gamma_{2i-1})\underline{\mathbf{u}} \sum_{j=0}^{i-1} |x^{j+1}| \\ &\quad + \underline{\mathbf{u}} \\ &\leqslant (1 + \gamma_{2(i+1)-1}) \sum_{j=1}^{i+1} |a_{n-(i+1)+j}||x^j| + (1 + \gamma_{2(i+1)-2})\underline{\mathbf{u}} \sum_{j=0}^{(i+1)-1} |x^j|. \end{aligned}$$

Par conséquent, on déduit que

$$\begin{aligned}
|s_{n-(i+1)}| &\leqslant (1+\mathbf{u})(|p_{n-(i+1)}| + |a_{n-(i+1)}|) \\
&\leqslant (1+\mathbf{u})(1+\gamma_{2(i+1)-1})\left[\sum_{j=1}^{i+1} |a_{n-(i+1)+j}||x^j| + |a_{n-(i+1)}|\right] \\
&\quad +(1+\mathbf{u})(1+\gamma_{2(i+1)-2})\underline{\mathbf{u}} \sum_{j=0}^{(i+1)-1} |x^j| \\
&\leqslant (1+\gamma_{2(i+1)})\sum_{j=0}^{i+1} |a_{n-(i+1)+j}||x^j| + (1+\gamma_{2(i+1)-1})\underline{\mathbf{u}} \sum_{j=0}^{(i+1)-1} |x^j|.
\end{aligned}$$

Les relations (3.19) and (3.20) sont donc vraies par récurrence. Ainsi, pour $i = 1, \ldots, n$,

$$\begin{aligned}
|p_{n-i}||x^{n-i}| &\leqslant (1+\gamma_{2n-1})\widetilde{p}(|x|) + (1+\gamma_{2n-2})\underline{\mathbf{u}}\sum_{j=0}^{n-1} |x^j|, \\
|s_{n-i}||x^{n-i}| &\leqslant (1+\gamma_{2n})\widetilde{p}(|x|) + (1+\gamma_{2n-1})\underline{\mathbf{u}}\sum_{j=0}^{n-1} |x^j|.
\end{aligned}$$

D'après la proposition 3.1 pour `TwoSum` et `TwoProd`, on a $|\pi_i| \leqslant \mathbf{u}|p_i| + 5\underline{\mathbf{u}}$ et $|\sigma_i| \leqslant \mathbf{u}|s_i|$ pour $i = 0, : n-1$. Par conséquent,

$$(\widetilde{p_\pi} + \widetilde{p_\sigma})(|x|) = \sum_{i=0}^{n-1}(|\pi_i| + |\sigma_i|)|x^i| \leqslant \mathbf{u}\sum_{i=1}^{n}(|p_{n-i}| + |\sigma_{n-i}|)|x^{n-i}| + 5\underline{\mathbf{u}}\sum_{i=0}^{n-1}|x^i|.$$

On obtient alors

$$\begin{aligned}
(\widetilde{p_\pi} + \widetilde{p_\sigma})(|x|) &\leqslant n\mathbf{u}\left[(2+\gamma_{2n-1}+\gamma_{2n})\widetilde{p}(|x|) + (2+\gamma_{2n-2}+\gamma_{2n-1})\underline{\mathbf{u}}\sum_{i=0}^{n-1}|x^i|\right] \\
&\quad +5\underline{\mathbf{u}}\sum_{i=0}^{n-1}|x^i| \\
&\leqslant 2n\mathbf{u}(1+\gamma_{2n})\widetilde{p}(|x|) + [5+2n\mathbf{u}(1+\gamma_{2n-1})]\,\underline{\mathbf{u}}\sum_{i=0}^{n-1}|x^i|
\end{aligned}$$

Puisque $2n\mathbf{u}(1+\gamma_{2n}) = \gamma_{2n}$ et $2n\mathbf{u}(1+\gamma_{2n-1}) \leqslant \gamma_{2n}$, on obtient finalement $(\widetilde{p_\pi}+\widetilde{p_\sigma})(|x|) \leqslant \gamma_{2n}\widetilde{p}(|x|) + (5+\gamma_{2n})\underline{\mathbf{u}}\sum_{i=0}^{n-1}|x^i|$. ■

Lemme 3.4. *Soit $p(x) = \sum_{i=0}^{n} a_i x^i$ un polynôme avec $a_i \in \mathbf{F}$, $0 \leqslant i \leqslant n$, $q(x) = \sum_{i=0}^{n} b_i x^i$ un polynôme avec $b_i \in \mathbf{F}$, $0 \leqslant i \leqslant n$ et $x \in \mathbb{F}$. Alors l'évaluation flottante de $r(x) = p(x) + q(x)$ selon l'algorithme suivant*

$r_n = \mathrm{fl}(a_n + b_n)$
for $i = n-1 : -1 : 0$
  $r_i = \mathrm{fl}(r_{i+1} \cdot x + (a_i + b_i))$
end
`res` $= r_0$

satisfait, en présence d'underflow,

$$|\texttt{res} - r(x)| \leqslant \gamma_{2n+1}\widetilde{r}(|x|) + (1+\gamma_{2n-1})\underline{\mathbf{u}}\sum_{i=0}^{n-1}|x|^i.$$

Démonstration. Nous avons $r_n = \mathrm{fl}(a_n + b_n) = (a_n + b_n)\langle 1\rangle$ et pour $i = n-1:0$,

$$r_i = \mathrm{fl}(r_{i+1}\cdot x + (a_i + b_i)) = \langle 2\rangle r_{i+1}x + \langle 2\rangle(a_i + b_i) + \eta_i, \quad \text{avec} \quad |\eta_i| \leqslant \underline{\mathbf{u}}.$$

On montre alors par récurrence que

$$r_0 = \langle 2n+1\rangle(a_n + b_n)x^n + \sum_{i=0}^{n-1}\langle 2(i+1)\rangle(a_i + b_i)x^i + \sum_{i=0}^{n-1}\langle 2i+1\rangle\eta_i x^i.$$

Puisque $r_0 = \mathrm{fl}((p+q)(x))$, on obtient finalement

$$|\texttt{res} - r(x)| \leqslant \gamma_{2n+1}(\widetilde{p}+\widetilde{q})(|x|) + (1+\gamma_{2n-1})\underline{\mathbf{u}}\sum_{i=0}^{n-1}|x^i|.$$

■

Théorème 3.2. *Soit* $p(x) = \sum_{i=0}^{n} a_i x^i$ *un polynôme avec* $a_i \in \mathbf{F}$, $0 \leqslant i \leqslant n$ *et* $x \in \mathbf{F}$. *Soit* `res` *le résultat calculé par l'algorithme 3.9. Alors, en présence d'underflow,*

$$|\texttt{res} - p(x)| \leqslant \mathbf{u}|p(x)| + \gamma_{2n}^2\widetilde{p}(|x|) + K\underline{\mathbf{u}}\sum_{i=0}^{n-1}|x|^i. \tag{3.21}$$

Démonstration. En considérant l'algorithme (3.9), on a $p(x) = s_0 + e(x) + 5\sum_{i=0}^{n-1}\eta_i x^i$. On en déduit que

$$\begin{aligned}
|\texttt{res} - p(x)| &= |(1+\varepsilon)(s_0 + \mathrm{fl}(e(x))) - p(x)| \\
&= |(1+\varepsilon)(s_0 + \mathrm{fl}(e(x)) - p(x)) + \varepsilon p(x)| \\
&= |(1+\varepsilon)(\mathrm{fl}(e(x)) - e(x) + 5\sum_{i=0}^{n-1}\eta_i x^i) + \varepsilon p(x)| \\
&\leqslant \mathbf{u}|p(x)| + (1+\mathbf{u})|\mathrm{fl}(e(x)) - e(x)| + 5(1+\mathbf{u})\underline{\mathbf{u}}\sum_{i=0}^{n-1}|x|^i.
\end{aligned}$$

En appliquant le lemme précédent, on obtient

$$|\,\mathrm{fl}(e(x)) - e(x)| \leqslant \gamma_{2n-1}(\widetilde{p_\pi}(|x|) + \widetilde{p_\sigma}(|x|)) + (1+\gamma_{2n-1})\underline{\mathbf{u}}\sum_{i=0}^{n-2}|x|^i.$$

De plus d'après le lemme précédent, on a

$$|\,\mathrm{fl}(e(x)) - e(x)| \leqslant \gamma_{2n-1}\gamma_{2n}\widetilde{p}(|x|) + \gamma_{2n-1}(5+\gamma_{2n})\underline{\mathbf{u}}\sum_{i=0}^{n-1}|x|^i + (1+\gamma_{2n-1}\underline{\mathbf{u}})\sum_{i=0}^{n-2}|x|^i.$$

Comme $(1+\mathbf{u})\gamma_{2n-1} \leqslant \gamma_{2n}$, on en déduit

$$|\mathtt{res} - p(x)| \leqslant \mathbf{u}|p(x)| + \gamma_{2n}^2\widetilde{p}(|x|) + \alpha$$

avec

$$\alpha = (1+\mathbf{u})[\gamma_{2n-1}(5+\gamma_{2n})+5]\underline{\mathbf{u}}\sum_{i=0}^{n-1}|x|^i + (1+\mathbf{u})(1+\gamma_{2n-1})\underline{\mathbf{u}}\sum_{i=0}^{n-2}|x|^i.$$

Comme $\underline{\mathbf{u}}$ est en général très petit, on peut borner α de la façon suivante

$$\alpha \leqslant (1+\mathbf{u})[6+6\gamma_{2n-1}+\gamma_{2n-1}\gamma_{2n}]\underline{\mathbf{u}}\sum_{i=0}^{n-1}|x|^i \leqslant K\underline{\mathbf{u}}\sum_{i=0}^{n-1}|x|^i,$$

avec $K \leqslant 7$. ■

3.7 Conclusion

Dans ce chapitre, nous avons présenté un algorithme précis et rapide pour l'évaluation de polynômes. Nous avons aussi donné une borne certifiée calculable en flottant de l'erreur commise lors de l'évaluation. Le code de l'algorithme peut être optimisé de manière très efficace si bien que cet algorithme est à la fois rapide en terme de flops mais aussi en terme de temps de calcul mesurés.

L'algorithme utilise seulement les opérations flottantes basiques que sont l'addition, la soustraction et la multiplication et utilise la même précision de calcul que celle des données. Cela offre la possibilité de l'insérer dans une librairie numérique puisqu'aucune architecture spéciale n'est requise.

Notre algorithme est basé sur les transformations exactes `TwoSum` et `TwoProduct` (algorithmes 3.1 et 3.4). Si ces fonctions étaient directement accessibles à partir du processeur, cela améliorait de façon significative la vitesse de l'algorithme. Une alternative possible serait l'utilisation des fonctions `TwoSumADD3` et `TwoProductFMA` (algorithmes 3.6 et 3.5).

Les transformations exactes permettent une analyse d'erreur plus simple. On a fréquemment utilisé l'égalité mathématique (3.4).

Chapitre 4

Applications de la sommation précise en géométrie algorithmique

4.1 Introduction

Les algorithmes qui utilisent des tests géométriques, tels que déterminer de quel coté d'une ligne se trouve un point, se trompent fréquemment à cause des erreurs d'arrondi. Une solution possible pour pallier ces problèmes est d'utiliser une arithmétique exacte hélas souvent très coûteuse en temps. Or, améliorer la vitesse des algorithmes de calcul géométrique robuste est un des challenges actuels [6, 26, 117], mais ces algorithmes actuels utilisent en entrée des entiers ou des rationnels. Ces algorithmes ne semblent pas appropriés quand on a besoin d'avoir des nombres flottants en entrée. Pour des entrées flottantes, Shewchuk [180] utilise une bibliothèque en précision arbitraire dans son implémentation en langage C de quatre prédicats géométriques, l'orientation 2D et 3D et l'appartenance à un cercle ou une sphère. Les entrées sont des flottants en simple ou double précision. La vitesse des algorithmes utilisés dans cette bibliothèque est due aux deux caractéristiques suivantes :

(i) des algorithmes rapides en précision arbitraire ;

(ii) une implémentation adaptative ; le temps d'exécution de l'algorithme dépend du degré d'incertitude sur le résultat.

Récemment, Demmel et Hida [52] ont montré que l'on peut transformer le calcul d'un déterminant de petite taille en le calcul d'une somme et ceci sans perdre d'information. Évaluer un déterminant précisément revient donc à calculer une somme précisément. Dans ce même article, ils ont proposé un algorithme de sommation précise utilisant un accumulateur long. Les nombres flottants p_i, $1 \leqslant i \leqslant n$, donné avec une précision de f bits pour la mantisse sont additionnés dans un accumulateur de F bits, $F > f$. Ils présentent différents algorithmes avec et sans tris sur les données. Ils donnent en particulier une analyse détaillée de la précision du résultat calculé en fonction de f,F et du nombre de termes à sommer. Il est clair que la sommation est une des opérations les plus utilisées en calcul scientifique et en particulier en algèbre linéaire. Beaucoup d'algorithmes ont été

développé pour calculer de telles sommations [4, 38, 51, 52, 105, 106, 127, 109, 111, 128, 130, 137, 140, 139, 153, 162, 163, 164, 165, 169, 180, 200]. Un excellent état de l'art sur ces différents algorithmes est présenté dans le chapitre 4 du livre [87].
Ces différents algorithmes de sommation partagent au moins un des inconvénients suivants :

- trier les données est nécessaire, soit en valeurs absolues soit par exposant,
- en plus de la précision courante, une précision étendue est nécessaire,
- un accès à la mantisse et/ou à l'exposant est nécessaire.

Chacune des propriétés que nous venons d'énumérer peut ralentir de façon significative les performances et restreindre les algorithmes à des architectures et des compilateurs spécifiques.

Dans ce chapitre, nous utilisons un algorithme récent de sommation dû à Ogita, Rump et Oishi [153] afin de proposer un algorithme précis et rapide de calcul de déterminants et de prédicats géométriques. L'avantage de cet algorithme est qu'il n'utilise qu'une seule précision courante et qu'il est adaptatif. Si le résultat a la précision désirée alors on termine l'algorithme. Si ce n'est pas le cas, on continue le calcul en augmentant à chaque fois la précision du résultat et ce tant que nous n'avons pas atteint la précision souhaité (en travaillant toujours avec la même précision dans les calculs). Contrairement à Demmel et Hida [52], nous avons pas besoin d'accumulateurs de grande précision et contrairement à Shewchuk [180], nous n'avons pas besoin de renormalisations qui ralentissent les calculs. Nous définissons aussi un conditionnement pour le calcul du déterminant et nous présentons ensuite une formule explicite pour l'évaluer.

Le reste du chapitre est organisé de la façon suivante. Dans la section 4.2, nous présentons l'algorithme de sommation d'Ogita, Rump et Oishi [153]. Cet algorithme permet de calculer de façon précise et rapide la somme de nombres flottants. Dans la section 4.3, nous présentons un algorithme de calcul du déterminant et nous l'appliquons au calcul de prédicats géométriques utilisés en géométrie algorithmique. Enfin, dans la section 4.4, nous présentons des expérimentations numériques de notre algorithme.

4.2 Sommation

Nous utilisons les notations de la section 3.2 (page 38) et 3.3 (page 40) du chapitre 3. Soient $p_i \in \mathbf{F}$, $1 \leqslant i \leqslant n$, n nombres flottants donnés. Le but de la section est de présenter un algorithme de sommation proposé par Ogita, Rump et Oishi [153] calculant une bonne approximation de la somme $s = \sum p_i$. Dans [153], ils cascadent l'algorithme `TwoSum` et somment les erreurs ainsi calculées afin d'améliorer la précision de la somme $\mathrm{fl}(\sum p_i)$. Leur algorithme est basé sur la transformation exacte `VecSum` qui transforme le vecteur p en un nouveau vecteur de somme identique mais avec un conditionnement amélioré d'un facteur $\mathbf{u}$.

Algorithme 4.1 (Ogita, Rump et Oishi [153, Algo. 4.3])**.** Transformations exacte pour la sommation.

function $p = \texttt{VecSum}(p)$
  for $i = 2 : n$
    $[p_i, p_{i-1}] = \texttt{TwoSum}(p_i, p_{i-1})$

Le vecteur p est transformé sans en changer la somme mais où p_n est remplacé par $\mathrm{fl}(\sum p_i)$. Les autres nouveaux p_i pour $1 \leqslant i \leqslant n-1$ sont les erreurs d'arrondis commises dans les calculs. C'est ce que Kahan a appelé un algorithme de distillation.

Nous décrivons maintenant l'algorithme de sommation corrigée.

Algorithme 4.2 (Ogita, Rump et Oishi [153, Algo. 4.4])**.** Sommation corrigée.

function $\texttt{res} = \texttt{Sum2}(p)$
  $p = \texttt{VecSum}(p)$
  $\texttt{res} = \mathrm{fl}\left(\left(\sum_{i=1}^{n-1} p_i\right) + p_n\right)$

L'algorithme `Sum2` satisfait la borne d'erreur suivante, qui signifie que le résultat calculé `res` est aussi précis que si la somme avait été calculée avec une précision double de la précision courante.

Proposition 4.1 (Ogita, Rump et Oishi [153, Prop. 4.5])**.** *Supposons que l'on applique l'algorithme 4.2 (*`Sum2`*) aux nombres flottants $p_i \in \mathbf{F}, 1 \leqslant i \leqslant n$. Posons $s := \sum p_i \in \mathbf{R}$ et $S := \sum |p_i|$ et supposons $n\mathbf{u} < 1$. Alors,*

$$|\texttt{res} - s| \leqslant \mathbf{u}|s| + \gamma_{n-1}^2 S. \tag{4.1}$$

La borne d'erreur (4.1) pour le résultat `res` de l'algorithme 4.2 n'est pas calculable en précision finie puisqu'elle fait intervenir la somme exacte s. Le théorème suivant [153, Cor. 4.7] calcule une borne d'erreur valide en précision finie avec arrondi au plus près, qui est aussi plus précise que (4.1).

Proposition 4.2 (Ogita, Rump et Oishi [153, Cor. 4.7])**.** *Soient $p_i \in \mathbf{F}, 1 \leqslant i \leqslant n$, n nombres flottants donnés. Si on ajoute*

if $2n\mathbf{u} \geqslant 1$, `error`("dimension too large"), end
$\beta = (2n\mathbf{u}/(1 - 2n\mathbf{u})) \cdot \left(\sum_{i=1}^{n-1} |p_i|\right)$
$\texttt{err} = \mathbf{u}|\texttt{res}| + (\beta + (2\mathbf{u}^2|\texttt{res}|))$

à l'algorithme 4.2 (`Sum2`*) et s'il n'y pas de message d'erreur, alors* `err` *satisfait*

$$\texttt{res} - \texttt{err} \leqslant \sum p_i \leqslant \texttt{res} + \texttt{err}.$$

Il peut être intéressant de cascader la transformation exacte `VecSum`. L'algorithme s'écrit alors comme suit.

Algorithme 4.3 (Ogita, Rump et Oishi [153, Algo. 4.8])**.** Sommation en précision K-uplée.

function $\texttt{res} = \texttt{SumK}(p, K)$
  for $k = 1 : K - 1$
    $p = \texttt{VecSum}(p)$
  $\texttt{res} = \mathrm{fl}\left(\left(\sum_{i=1}^{n-1} p_i\right) + p_n\right)$

Le théorème suivant donne une estimation de l'erreur dans l'algorithme 4.3 qui signifie que le résultat `res` est aussi précis que s'il avait été calculé avec une précision de K-fois la précision courante.

Proposition 4.3 (Ogita, Rump et Oishi [153, Prop. 4.10])**.** *Soient $p_i \in \mathbf{F}, 1 \leqslant i \leqslant n$, n nombres flottants donnés et supposons $4n\mathbf{u} \leqslant 1$. Alors le résultat* `res` *de l'algorithme 4.3 (*`SumK`*) vérifie pour $K \geqslant 3$*

$$|\texttt{res} - s| \leqslant (\mathbf{u} + 3\gamma_{n-1}^2)|s| + \gamma_{2n-2}^K S,$$

avec $s := \sum p_i$ et $S := \sum |p_i|$.

La proposition suivante donne une borne d'erreur valide en arithmétique flottante avec arrondi au plus près pour l'algorithme de sommation classique.

Proposition 4.4. *Soient $p_i \in \mathbf{F}$, $1 \leqslant i \leqslant n$, n nombres flottants données. Supposons que $2n\mathbf{u} \leqslant 1$. Alors le résultat $\texttt{res} = \mathrm{fl}(\sum p_i)$ vérifie*

$$|\texttt{res} - s| \leqslant \mathrm{fl}(\gamma_{2n} \sum |p_i|).$$

Démonstration. On montre dans [87, Lem. 8.4] que

$$|\texttt{res} - s| \leqslant \gamma_{n-1} \sum |p_i|.$$

Si $m\mathbf{u} \leqslant 1$ pour $m \in \mathbf{N}$, on a $\mathrm{fl}(m\mathbf{u}) = m\mathbf{u}$ et $\mathrm{fl}(1 - m\mathbf{u}) = 1 - m\mathbf{u}$. Il en résulte que

$$\gamma_m \leqslant (1 - \mathbf{u})^{-1} \mathrm{fl}(\gamma_m).$$

De plus, une récurrence simple montre que

$$\sum |p_i| \leqslant (1 + \mathbf{u})^{n-1} \mathrm{fl}(\sum |p_i|).$$

Il vient alors

$$\begin{aligned}
|\texttt{res} - s| &\leqslant \gamma_{n-1} \sum |p_i| \\
&\leqslant \gamma_{n-1}(1 + \mathbf{u})^{n-1} \mathrm{fl}(\sum |p_i|) \\
&\leqslant \gamma_{2n-2} \mathrm{fl}(\sum |p_i|) \quad \text{puisque} \quad (1 + \mathbf{u})\gamma_n \leqslant \gamma_{n+1} \\
&\leqslant (1 - \mathbf{u})^2 \gamma_{2n} \mathrm{fl}(\sum |p_i|) \quad \text{puisque} \quad \gamma_n \leqslant (1 - \mathbf{u})\gamma_{n+1} \\
&\leqslant (1 - \mathbf{u}) \mathrm{fl}(\gamma_{2n}) \mathrm{fl}(\sum |p_i|) \\
&\leqslant \mathrm{fl}(\gamma_{2n} \sum |p_i|).
\end{aligned}$$

Cela termine la preuve. ■

4.3 Applications en géométrie algorithmique

4.3.1 Calcul précis de déterminants 2×2 et 3×3

La littérature sur le calcul précis de déterminant est assez importante (voir par exemple [38, 25, 42] et leurs références). La plupart des algorithmes utilise des bibliothèques multiprécision pour calculer de façon précise le déterminant. Nous présentons ci-après un algorithme de calcul de déterminant avec une précision relative ε fixée à l'avance en utilisant seulement une seule précision courante (ici la double précision IEEE 754). Regardons le cas d'un déterminant 2×2

$$\text{det2} = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

En utilisant `TwoProduct`, on peut écrire

$$[x, y] = \texttt{TwoProduct}(a, d) \quad \text{et} \quad [z, t] = \texttt{TwoProduct}(b, c).$$

On a alors $\text{det2} = x + y + z + t$. Nous avons transformé le problème du calcul de déterminant en un problème de sommation. On peut alors appliquer l'algorithme de sommation précise 4.2 pour calculer cette somme.

On peut faire de même avec un déterminant 3×3

$$\text{det3} = \begin{vmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{vmatrix} = \sum_{\sigma \in \mathfrak{S}_3} \text{signature}(\sigma) a_{1,\sigma(1)} \cdot a_{2,\sigma(2)} \cdot a_{3,\sigma(3)},$$

en utilisant la transformation exacte `ThreeProduct` ci-dessous pour transformer $a_{1,\sigma(1)} \cdot a_{2,\sigma(2)} \cdot a_{3,\sigma(3)}$ en une somme de quatre nombres flottants.

Algorithme 4.4. Transformation exacte pour le produit de trois nombres flottants.

function $[x, y, z, t] = \texttt{ThreeProduct}(a, b, c)$
$[p, e] = \texttt{TwoProduct}(a, b)$
$[x, y] = \texttt{TwoProduct}(p, c)$
$[z, t] = \texttt{TwoProduct}(e, c)$

Étant donnés $a, b, c \in \mathbf{F}$ trois nombres flottants, `ThreeProduct` transforme le produit abc en une somme non évaluée $x + y + z + t$ de quatre nombres flottants. En effet, on peut écrire

$$abc = (p + e)c = pc + ec = x + y + z + t.$$

Nievergelt [147] a montré que l'on peut utiliser cela pour calculer de façon précise l'aire d'un triangle ou le volume d'un tétraèdre.

L'algorithme suivant calcule le déterminant d'une matrice avec une erreur relative inférieure à ε donné. Nous appelons `DetVector` la fonction qui transforme le calcul du déterminant en le calcul d'une somme (de la même façon que précédemment). Nous calculons alors la somme et une borne d'erreur. Si l'erreur est inférieure à la précision voulue ε alors on s'arrête. Sinon on réitère la sommation.

Algorithme 4.5. Algorithme de calcul de déterminant avec une erreur relative ε.

function $\texttt{resdet} = \texttt{det}(A, \varepsilon)$
  $p = \texttt{DetVector}(A)$
  repeat
    $p = \texttt{VecSum}(p)$
    $\text{res} = p_n$
    $\beta = (2n\mathbf{u}/(1-2n\mathbf{u})) \cdot \left(\sum_{i=1}^{n-1} |p_i|\right)$
    $\texttt{err} = \mathbf{u}|\texttt{res}| + (\beta + (2\mathbf{u}^2|\texttt{res}|))$
  until $(\texttt{err} \leqslant \varepsilon|\texttt{res}|)$
  $\texttt{resdet} = p_n$

4.3.2 Prédicats géométriques robustes

Une application requérant des résultats garantis est par exemple le calcul de prédicats géométriques. Des algorithmes tels que la triangulation de Delaunay et la génération de maillages ont besoin de tests géométriques robustes. Shewchuk [180] donne un panorama des problèmes de ce type et présente une arithmétique flottante adaptative (et arbitraire) pour résoudre ces problèmes. La plupart des prédicats géométriques requière le calcul du signe d'un déterminant de petite taille. Des travaux récents sur ce sujet sont [180, 6, 26, 117, 52]. Considérons par exemple le prédicat ORIENT3D qui détermine si un point D se situe à droite ou à gauche d'un plan orienté défini par les points A, B et C. Le résultat du prédicat dépend du signe du déterminant

$$\text{ORIENT3D}(a,b,c,d) = \text{sign}\begin{vmatrix} a_x & a_y & a_z & 1 \\ b_x & b_y & b_z & 1 \\ c_x & c_y & c_z & 1 \\ d_x & d_y & d_z & 1 \end{vmatrix} \tag{4.2}$$

$$= \text{sign}\begin{vmatrix} a_x - d_x & a_y - d_y & a_z - d_z \\ b_x - d_x & b_y - d_y & b_z - d_z \\ c_x - d_x & c_y - d_y & c_z - d_z \end{vmatrix}. \tag{4.3}$$

Le résultat calculé est de même signe que le résultat exact si l'erreur relative est inférieure à un. Par conséquent, il suffit de calculer une approximation du déterminant avec une erreur relative inférieure à un. Demmel et Hida [52] ont récemment proposé une méthode pour certifier le signe d'un déterminant de petite taille. Pour ce faire, ils utilisent un algorithme de sommation précise avec des accumulateurs longs. Dans notre cas, nous utilisons les mêmes techniques que dans la section précédente. Tout d'abord, nous calculons le déterminant en utilisant la formule (4.3). On calcule aussi une borne certifié de l'erreur. Si cette erreur relative est inférieure à un alors on a obtenu le signe du déterminant.

Sinon, on transforme le déterminant en une somme de 96 nombres flottants : en effet, le déterminant peut s'exprimer comme une somme de 24 monômes de la forme $\pm a_i b_j c_k$, et chacun de ces monômes peut être transformé en la somme de 4 nombres flottants avec `ThreeProduct` (dans notre implémentation, on utilise une version un peu améliorée de ce procédé en évitant un certain nombre de calculs redondants). On peut alors appliquer

l'algorithme de sommation 4.3 jusqu'à obtenir une erreur relative plus petite que un. Ceci peut être fait en utilisant la borne d'erreur de la proposition 4.2. Dans l'algorithme suivant, nous supposons toujours que nous avons une fonction `DetVector` qui transforme la matrice dont nous voulons calculer le déterminant en un vecteur dont la somme est le déterminant (ceci en utilisant `ThreeProduct`). On note n la longueur du vecteur (dans notre cas $n = 94$).

Algorithme 4.6. Algorithme de calcul du prédicat ORIENT3D(a,b,c,d).

function sign = `Orient3D`(A)
  res = `det`(A)
  err = `ErrDet`(A)
  if (err $\leqslant$ |res|)
    return `sign` = sign(res)
  p = `DetVector`(A)
  repeat
    p = `VecSum`(p)
    res = p_n
    $\beta = (2n\mathbf{u}/(1-2n\mathbf{u})) \cdot \left(\sum_{i=1}^{n-1} |p_i|\right)$
    err = $\mathbf{u}|\text{res}| + (\beta + (2\mathbf{u}^2|\text{res}|))$
  until (err $\leqslant$ |res|)
  `sign` = sign(res)

Dans l'algorithme 4.6, la fonction `det` calcule le déterminant en utilisant la formule (4.3). La fonction `ErrDet` calcule quant à elle une borne d'erreur certifiée sur le calcul de `det` en utilisant des techniques similaires à celles utilisées dans la proposition 4.4.

4.3.3 Conditionnements pour les déterminants

Tout d'abord, nous définissons un *conditionnement global* pour le problème du calcul du déterminant d'une matrice donnée par

$$\kappa(A) := \limsup_{\varepsilon \to 0} \left\{ \frac{|\det(A+H) - \det(A)|}{\varepsilon |\det(A)|} : \|H\|_F \leqslant \varepsilon \|A\|_F \right\}.$$

Nous utilisons la norme de Frobenius définie par

$$\|A\|_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{i,j}|^2 \right)^{1/2} = \operatorname{trace}(A^* A)^{1/2}.$$

Le choix de cette norme plutôt que la norme 2 est motivé par le fait que l'on peut obtenir une expression explicite pour le conditionnement.

Théorème 4.1. *Soit $A \in \mathbf{R}^{n \times n}$ une matrice réelle. Alors*

$$\kappa(A) = \frac{\|A\|_F \|\operatorname{Com}(A)\|_F}{|\det(A)|},$$

ou $\operatorname{Com}(A) := ((-1)^{i+j} \det(A_{ij}))$ *est la comatrice avec A_{ij} représentant la sous-matrice de A obtenue en supprimant la ligne i et la colonne j.*

Démonstration. La fonction $\det : \mathbf{R}^{n\times n} \to \mathbf{R}, A \mapsto \det(A)$ est Fréchet différentiable puisque c'est un polynôme en les coefficients de la matrice. Notons (E_{ij}) la base canonique de $\mathbf{R}^{n\times n}$. Soit $A = (a_{ij})_{1\leqslant i,j\leqslant n}$ une matrice donnée. On veut calculer $\frac{\partial \det}{\partial a_{ij}}(A)$. Soit $C \in \mathbf{R}^{n\times n}$ la comatrice de A. La n-linéarité du déterminant implique que, pour tout (i,j) et pour tout $t \in \mathbf{R}$,

$$\det(A + tE_{ij}) = \det(A) + tC_{ij}$$

et donc

$$\frac{\partial \det}{\partial a_{ij}}(A) = \lim_{t\to 0} \frac{\det(A + tE_{ij}) - \det(A)}{t} = C_{ij}.$$

Ainsi, si $H = (h_{ij}) \in \mathbf{R}^{n\times n}$, on a

$$D\det(A)(H) = \sum_{i,j} h_{ij} \frac{\partial \det}{\partial a_{ij}}(A) = \sum_{i,j} h_{ij} C_{ij} = \operatorname{trace}(C^T H).$$

Ici D représente la différentielle. On déduit alors de la définition du conditionnement que

$$\kappa(A) = \frac{\|A\|_F \|D\det(A)\|}{|\det(A)|}.$$

Par définition, on a aussi

$$\|D\det(A)\| = \sup_{\|M\|_F=1} |D\det(A)(M)| = \sup_{\|M\|_F=1} |\operatorname{trace}(C^T M)|.$$

Comme $\langle A, B\rangle = \operatorname{trace}(A^T B)$ est un produit scalaire sur $\mathbf{R}^{n\times n}$ associé à la norme de Frobenius, on déduit facilement de l'inégalité de Cauchy-Schwarz que

$$\|D\det(A)\| = \|C\|_F.$$

Par conséquent, $\kappa(A) = \|A\|_F \|\operatorname{Com}(A)\|_F / |\det(A)|$. Si A est inversible, alors on a $C^T = \det(A)A^{-1}$ et donc $\kappa(A) = \|A\|_F \|A^{-1}\|_F$. ∎

On peut aussi définir un *conditionnement par composante* pour le déterminant par

$$\operatorname{cond}(A) := \lim_{\varepsilon\to 0} \sup \left\{ \frac{|\det(A+H) - \det(A)|}{\varepsilon |\det(A)|} : |H| \leqslant \varepsilon |A| \right\},$$

ou les valeurs absolues et les comparaisons sont faites composantes par composantes. Un raisonnement analogue à celui fait pour le produit scalaire [153] montre que

$$\operatorname{cond}(A) = \frac{n \operatorname{per}(|A|)}{\det(A)},$$

ou $\operatorname{per}(A)$ désigne le permanent de la matrice A définie par

$$\operatorname{per}(A) = \sum_{\sigma\in\mathfrak{S}_n} \prod a_{i,\sigma(i)}.$$

Ce conditionnement va nous permettre d'apprécier la difficulté de calculer un déterminant. Cela va nous permettre dans la section suivante de montrer l'efficacité de notre algorithme pour des conditionnements élevés.

On trouvera une autre définition pour le conditionnement du déterminant dans [86, Pb 13.15].

4.4 Résultats expérimentaux

Dans cette section, nous comparons les trois algorithmes suivants.

1. APPROXIMATE. L'algorithme direct en double précision qui n'est pas robuste.
2. ADAPTIVE. L'algorithme de Shewchuk en multiprécision adaptative [180]. Il fonctionne en obtenant successivement une meilleure estimation du déterminant et s'arrête quand l'erreur relative est inférieure à un. Cet algorithme est considéré comme le meilleur algorithme pour le calcul des prédicats géométriques.
3. ORIENT3DEXACT. Notre algorithme 4.6 qui est lui aussi adaptatif.

TAB. 4.1 – Comparaison des 3 algorithmes. Le ratio est le temps de calcul comparé à celui de l'algorithme ADAPTIVE. Tous les temps sont en μs.

Algorithmes	Points			
	Aléatoire		Presque coplanaire	
	Temps	Ratio	Temps	Ratio
APPROXIMATE	0.023	0.34	0.023	0.0042
ADAPTIVE	0.066	1	5.4	1
ORIENT3DEXACT	0.066	1	2.4	0.44

Le tableau 4.1 donne les performances des trois algorithmes. Le ratio est le temps de calcul des différents algorithmes comparés avec celui de l'algorithme ADAPTIVE. Toutes les mesures de temps ont été faites en C sur un Intel Pentium IV 3 Ghz avec le compilateur GNU C (`gcc-3.4.1`). Chaque temps reporté est une moyenne des temps sur 1000 tests. Pour des points choisis de manière aléatoire, la probabilité qu'ils soient presque coplanaires est très faible. Par conséquent, les trois algorithmes partagent des temps de calcul assez proches. En effet, les trois algorithmes utilisent la même méthode de calcul (le calcul du déterminant par la formule (4.3)) pour calculer le déterminant. Notre algorithme ORIENT3DEXACT et l'algorithme de Shewchuk ADAPTIVE calculent aussi une borne d'erreur ce qui explique que ces deux algorithmes soient presque deux fois plus longs que l'algorithme classique ; c'est le prix à payer pour la robustesse.

Pour des points presque coplanaires, notre algorithme est plus de deux fois plus rapide que l'algorithme ADAPTIVE de Shewchuk. Cela s'explique en partie par le fait que Shewchuk a besoin de renormaliser ses expansions ce qui nécessite un tri des données. Pour nos choix de points presque coplanaires, le conditionnement par composante varie de 10^{10} jusqu'à 10^{50}. Les points presque coplanaires ne représentent qu'une partie infime des entrées dans la pratique. Néanmoins, cela représente les cas les plus coûteux en temps. Il est donc important d'être capable de les traiter le plus efficacement possible.

4.5 Conclusion

Dans ce chapitre, nous avons proposé un algorithme rapide et robuste pour le calcul du prédicat géométrique ORIENT3D. Notre algorithme est aussi rapide que celui de

Shewchuk dans les cas bien conditionnés et deux fois plus rapide dans les cas très mal conditionnés. Nous n'avons testé que le prédicat ORIENT3D. Bien entendu, les mêmes techniques s'appliquent aisément pour les autres prédicats comme INCIRCLE et INSPHERE par exemple (voir [180]). Le principal potentiel de notre algorithme est de fournir une méthode simple et efficace pour augmenter la précision des variables critiques dans les algorithmes numériques. Les techniques que nous avons utilisées sont suffisamment simples pour être implémentées facilement sur n'importe quelle architecture.

Deuxième partie

Pseudozéros, conditionnement et applications

Chapitre 5

Présentation des pseudozéros

5.1 Introduction

L'étude des racines d'un polynôme est un problème récurrent en mathématiques et qui trouve son origine dans l'antiquité (voir [160] pour un historique). En effet, la connaissance des racines permet entre autre la factorisation du polynôme ou bien le calcul du PGCD de deux polynômes. Or Galois ayant montré que pour des polynômes de degré supérieur ou égal à 5, on ne peut trouver les racines par radicaux, un calcul numérique s'impose alors afin de déterminer une approximation de ces racines. Le problème de la précision finie en arithmétique des ordinateurs intervient alors. En effet, les coefficients étant codés comme des nombres flottants, il y a une perturbation des coefficients en passant en arithmétique flottante. Celle-ci peut influer de manière importante sur les racines du polynôme.

Trois notions permettent d'étudier l'influence de la précision finie sur les racines :

- le nombre de conditionnement et la théorie des perturbations linéaires [87, 36] ;
- les pseudozéros [36, 54, 143, 185, 192, 206] ;
- le pseudospectre de la matrice compagnon [192, 194], notion issue plus particulièrement de l'algèbre linéaire.

L'intérêt des pseudozéros vient du fait qu'ils permettent d'étudier de manière géométrique la sensibilité des racines du polynôme par rapport aux perturbations de ses coefficients et donc se prêtent bien à une représentation graphique.

Dans la section suivante, nous introduisons les premières notations et étudions les différentes mesures sur les perturbations. Puis, nous définissons de façon rigoureuse la notion de pseudozéros pour énoncer quelques propriétés. Ensuite, nous exposons l'algorithme que nous avons utilisé pour calculer ces ensembles et énonçons quelques théorèmes relatifs au nombre de racines dans les composantes connexes d'un ensemble de pseudozéros et à la des multiplicités des racines. Enfin, nous proposons quelques simulations numériques.

5.2 Les perturbations

5.2.1 Choix du type de perturbations

Soit à trouver les racines du polynôme $p \in \mathcal{P}_n = \mathbf{C}_n[z]$ (ensemble des polynômes à coefficients dans $\mathbf{C}$ de degré au plus n)

$$p(z) = p_n z^n + \cdots + p_1 z + p_0. \tag{5.1}$$

Soit $\|\cdot\|$ une norme sur $\mathcal{P}_n$, on peut alors définir sur $\mathcal{P}_n$ une topologie en définissant le ε-voisinage de p par

$$N_\varepsilon(p) = \{\widehat{p} \in \mathcal{P}_n : \|p - \widehat{p}\| \leqslant \varepsilon\}.$$

On parle alors de perturbations non structurées. Nous discuterons plus loin les différents choix de norme pour $\|\cdot\|$.

Remarque. On peut affaiblir cette définition en prenant non plus une norme sur $\mathcal{P}_n$ mais une distance.

Notons $Z(p)$ l'ensemble des racines de p et pour tout $\varepsilon > 0$, définissons l'ensemble des ε-pseudozéros par

$$Z_\varepsilon(p) = \{z \in \mathbf{C} : z \in Z(\widehat{p}) \text{ pour } \widehat{p} \in N_\varepsilon(p)\}.$$

De tels ensembles ont été étudiés par Mosier [143] puis par Trefethen et Toh [192].

On considère en fait que le polynôme $\widehat{p}$ est le polynôme dont les coefficients sont des perturbations de ceux de p. Par conséquent, pour définir $\widehat{p}$, il nous faut définir les perturbations des coefficients. On distingue classiquement deux types de perturbations :

– les perturbations non structurées,
– les perturbations structurées.

Les perturbations structurées permettent de choisir la forme du polynôme perturbé. Un type générique est

$$\widehat{p}(z) = \sum_{i=0}^{n} (p_i + \varepsilon f_i(\varepsilon)) z^i,$$

où les $(p_i)_{i=0,\ldots,n}$ sont les coefficients de p et où les fonctions f_i appartiennent à $\mathbf{C}[z]$.

Un type plus simple de perturbations structurées sont les perturbations dites *linéaires* de la forme

$$\widehat{p}(z) = p(z) + \varepsilon g(z),$$

où g est un polynôme de degré inférieur ou égal à celui de p donnant la structure de la perturbation.

En suivant [36], on distingue surtout deux types de perturbations non structurées :

– les perturbations *globales*,
– les perturbations *par composante*.

5.2.1.1 Les perturbations *globales*

Soit p défini par la relation (7.1) et $\widehat{p}$ un polynôme perturbé de p. On définit la norme *globale* par

$$\|p-\widehat{p}\|^{\mathcal{N}} = \frac{\|p-\widehat{p}\|}{\beta},$$

où $\|\cdot\|$ est une norme sur les polynômes et β est un réel quelconque. On prendra souvent $\beta = \|p\|$ afin d'avoir une norme relative.

Remarque. Un *scaling* (cf [36, p. 51]) sur la norme est utilisé dans [192]. Cela consiste à définir la norme $\|x\|_d = \|Dx\|_2$ ou D est une matrice diagonale $(d_0, \ldots, d_n)$. Dans ce cas $\|p\|_d = \left(\sum_{i=0}^n |d_i|^2|a_i|^2\right)^{1/2}$. Cela revient à prendre la norme *globale* avec $\beta = 1$ et $\|\cdot\| = \|\cdot\|_d$.

5.2.1.2 Les perturbations *par composante*

En utilisant les mêmes notations que précédemment, on définit la norme *par composante* par

$$\|p-\widehat{p}\|^{\mathcal{C}} = \max_i \frac{|p_i-\widehat{p}_i|}{f_i},$$

où les $(f_i)_{i=0,\ldots,n}$ sont des réels positifs. En général, on prend $f_i = |p_i|$ afin d'avoir une norme relative.

Remarque. Il s'agit de la norme utilisée par Mosier [143].

En conclusion, les perturbations *par composante* permettent de « modéliser » les perturbations des coefficients alors que les perturbations *globales* ne décrivent que globalement la perturbation du vecteur des coefficients.

5.3 Étude et calcul des ε-pseudozéros

Après avoir défini les types de perturbations utilisées et les normes associées, nous pouvons définir rigoureusement la notion de ε-pseudozéros. On rappelle que l'on définit le ε-voisinage de p par

$$N_\varepsilon^{\mathcal{T}}(p) = \left\{\widehat{p} \in \mathcal{P}_n : \|p-\widehat{p}\|^{\mathcal{T}} \leqslant \varepsilon\right\}, \tag{5.2}$$

où $\mathcal{T}$ désigne soit $\mathcal{C}$ pour la norme *par composante*, soit $\mathcal{N}$ pour la norme *globale*. On peut alors définir l'ensemble des ε-pseudozéros par

$$Z_\varepsilon^{\mathcal{T}}(p) = \left\{z \in \mathbf{C} : \widehat{p}(z) = 0 \text{ pour } \widehat{p} \in N_\varepsilon^{\mathcal{T}}(p)\right\}. \tag{5.3}$$

Cette définition ne se prête pas au calcul car elle n'est pas constructive. Néanmoins, selon le type de perturbations (*globales* ou *par composante*), nous identifions une forme calculable de l'ensemble des ε-pseudozéros.

Dans le cas de perturbations *globales*, on a le théorème suivant (voir [36]).

Théorème 5.1. *L'ensemble des pseudozéros en norme* globale *vérifie*

$$Z_\varepsilon^{\mathcal{N}}(p) = \left\{ z \in \mathbf{C} : \frac{|p(z)|}{\|\underline{z}\|_* \beta} \leqslant \varepsilon \right\}, \tag{5.4}$$

où $\underline{z} = (1, z, \ldots, z^n)$ *et* $\|\cdot\|_*$ *est la norme duale de* $\|\cdot\|$.

Mosier [143] utilise déjà cet ensemble avec la norme ∞. De la même façon, Trefethen et Toh [192] réutilisent cet ensemble avec la norme 2. Ici, nous généralisons leur résultat pour une norme arbitraire.

Remarque. Nous identifierons par la suite $\mathcal{P}_n$ et $\mathbf{C}^{n+1}$. On peut même, si besoin est, identifier $\mathcal{P}_n$ et $\mathbf{C}^n$ en considérant les polynômes unitaires (coefficient dominant p_n égal à 1). Un polynôme sera donc vu soit comme un polynôme au sens classique du terme soit comme le vecteur de ses coefficients.

Démonstration. On rappelle que la norme duale $\|\cdot\|_*$ sur $\mathbf{C}^{n+1}$ est définie par

$$\|x\|_* = \max_{z \neq 0} \frac{|z^T x|}{\|z\|} = \max_{\|z\|=1} |z^T x|.$$

Si $z \in Z_\varepsilon(p)$ alors il existe $\widehat{p} \in \mathcal{P}_n$ tels que $\widehat{p}(z) = 0$ et $\|p - \widehat{p}\|^{\mathcal{N}} = \|p - \widehat{p}\|/\beta \leqslant \varepsilon$. En utilisant l'inégalité de Hölder généralisée (*i.e.* $|x^T y| \leqslant \|x\|\|y\|_*$), on a

$$|p(z)| = |p(z) - \widehat{p}(z)| = |\sum_{i=0}^{n} (p_i - \widehat{p}_i) z^i| \leqslant \|p - \widehat{p}\| \|\underline{z}\|_*.$$

Et par conséquent, on a bien $|p(z)| \leqslant \varepsilon \|\underline{z}\|_* \beta$.

Pour montrer la réciproque, soit $u \in \mathbf{C}$ tel que $|p(u)| \leqslant \varepsilon \|\underline{u}\|_* \beta$. Un résultat classique nous permet d'affirmer l'existence d'un vecteur $d = (d_i) \in \mathbf{C}^{n+1}$ de norme 1 vérifiant $d^T \underline{u} = \|\underline{u}\|_*$ ([87, p. 119] ou [99, p. 278]). Ce vecteur d est appelé le vecteur dual de $\underline{u}$. Définissons le polynôme r par

$$r(z) = \sum_{k=0}^{n} d_k z^k \text{ avec } r_k = d_k,$$

et le polynôme p_u par

$$p_u(z) = p(z) - \frac{p(u)}{r(u)} r(z). \tag{5.5}$$

On peut vérifier que p_u est le polynôme le plus proche de p (au sens de la norme $\|\cdot\|$) ayant u comme racine.
Il est clair que $r(u) = d^T \underline{u} = \|\underline{u}\|_*$ et $p_u(u) = 0$. Donc

$$\|p - p_u\| = \frac{|p(u)|}{|r(u)|} \|r\| \leqslant \|d\| \varepsilon \beta.$$

Comme $\|d\| = 1$, on a

$$\|p - p_u\| \leqslant \varepsilon \beta.$$

Ainsi $u \in Z_\varepsilon(p)$. ■

Norme	Norme duale
$\|x\|_1 := \sum_j \|x_j\|$	$\|x\|_1^* = \max_j \|x_j\| = \|x\|_\infty$
$\|x\|_2 := (\sum_j \|x_j\|^2)^{1/2}$	$\|x\|_2^* = (\sum_j \|x_j\|^2)^{1/2} = \|x\|_2$
$\|x\|_\infty := \max_j \|x_j\|$	$\|x\|_\infty^* = \sum_j \|x_j\| = \|x\|_1$

TAB. 5.1 – Normes duales pour les normes usuelles sur $\mathbf{K}^N$ avec $\mathbf{K} = \mathbf{R}$ ou $\mathbf{C}$

Le tableau 9.1 donne les normes duales pour les normes usuelles

De la même façon, pour les perturbations *par composante*, nous avons le théorème suivant.

Théorème 5.2. *L'ensemble des pseudozéros en norme* par composante *vérifie*

$$Z_\varepsilon^{\mathcal{C}}(p) = \left\{ z \in \mathbf{C} : \frac{|p(z)|}{\sum_{i=0}^n |f_i||z|^i} \leqslant \varepsilon \right\}. \tag{5.6}$$

Remarque. Dans les deux théorèmes précédents, la forme générale de $Z_\varepsilon(p)$ est $Z_\varepsilon(p) = \{z \in \mathbf{C} : |g(z)| \leqslant \varepsilon\}$ avec g une fonction calculable. Dans la suite nous utiliserons g afin d'encapsuler les deux cas correspondants aux perturbations *globales* et *par composante*.

Pour un lien entre le conditionnement de l'évaluation polynomiale et les pseudozéros, on pourra se reporter à la sous-section 2.1.3 du chapitre 2.

À partir de ces théorèmes, nous pouvons alors calculer les pseudozéros. Pour cela, nous avons utilisé le logiciel MATLAB. L'algorithme utilisé est l'algorithme 5.1.

Algorithme 5.1 Calcul de pseudozéros

Entrée : le polynôme p et la perturbation ε

Sortie : le pseudozéro tracé dans le plan complexe

1: On maille un carré contenant toutes les racines de p à l'aide de la commande MATLAB `meshgrid`.
2: On calcule $g(z)$ pour tous les points z de la grille.
3: On affiche la ligne de niveau $|g(z)| = \varepsilon$ à l'aide de la commande MATLAB `contour`.

L'arrêt et la correction de l'algorithme est immédiat.

Étudions maintenant la complexité de cet algorithme. Notons L la longueur du carré et h le pas de discrétisation. L'évaluation de $g(u)$ nécessite l'évaluation d'un polynôme, ce qui se fait en $\mathcal{O}(n)$, le calcul de la norme d'un vecteur, dont la complexité dépend de la norme. Notons $\mathcal{O}(\|\cdot\|_*)$ cette complexité. La complexité de l'algorithme précédent est donc en $\mathcal{O}((L/h)^2(n + \|\cdot\|_*))$.

Nous avons testé notre programme sur les polynômes suivants qui sont utilisés dans les articles [143, 192, 206].

- le polynôme de Wilkinson $W_{20}(z) = \Pi_{k=1}^{20}(z-k)$;
- le polynôme de Wilkinson tronqué $W_{10}(z) = \Pi_{k=1}^{10}(z-k)$;

– le polynôme $E(z) = \sum_{k=0}^{20} z^k/k!$;
– le polynôme $U(z) = z^{20} + z^{19} + \cdots + z + 1$;
– le polynôme unitaire ayant comme zéros $2^{-10}, 2^{-9}, \ldots, 2^9$.

La figure 5.1 représente l'ensemble des pseudozéros du polynôme de Wilkinson W_{20} en norme *par composante* en ne perturbant que le coefficient de z^{19} avec une perturbation $\varepsilon = 2^{-23}$ (voir [198] ou [199]).

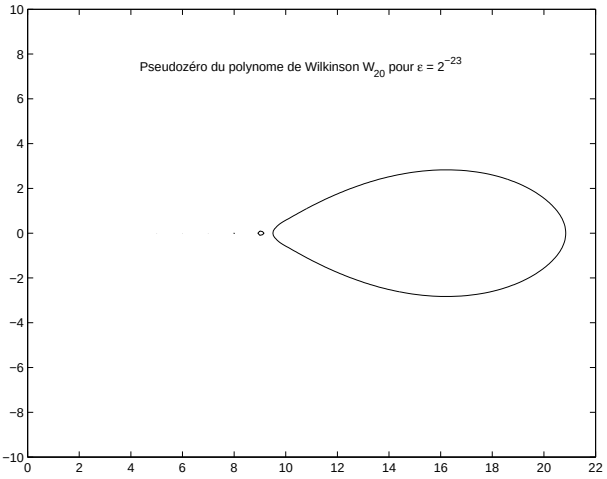


FIG. 5.1 – Pseudozéros du polynôme de Wilkinson W_{20} pour $\varepsilon = 2^{-23}$

Remarque. Nous effectuons ici quelques remarques concernant l'effet de la *précision finie* sur les pseudozéros. En effet, deux questions se posent principalement. D'une part, comment peut-on définir la grille *a priori* de façon à ce qu'elle contienne toutes les racines et que sa finesse permette une bonne précision des pseudozéros ? D'autre part, le calcul des pseudozéros revient à évaluer un polynôme. Or naturellement, cette évaluation est entachée d'une erreur. Quelles sont alors les conséquences de cette erreur sur les pseudozéros ?

Il semble intéressant donc de diminuer la taille de la grille en localisant de manière plus simple les racines. Soit p un polynôme unitaire de degré n et $\{z_i\}$ l'ensemble de ses n racines. Notons $r = \max_{i=1,\ldots,n} |z_i|$. On peut alors montrer [138, p. 154] que

$$r \leqslant \max\{1, \sum_{k=1}^{n} |p_k|\}.$$

Cela nous permet de définir un carré contenant l'ensemble des pseudozéros. En effet, soit $z \in Z_\varepsilon(p)$. On sait alors qu'il existe $\widehat{p} \in N_\varepsilon(p)$ tel que $\widehat{p}(z) = 0$. Le complexe z étant une racine de $\widehat{p}$, il est donc inférieur à la plus grande racine de $\widehat{p}$. Par conséquent, on a l'inégalité

$$|z| \leqslant \max\{1, \sum_{k=1}^{n} |\widehat{p}_k|\}.$$

Supposons que l'on soit dans le cas d'une perturbation en norme p de Hölder $\|\cdot\|_p$. On sait alors que $\|p - \widehat{p}\|_p \leqslant \varepsilon$. Or $\|\cdot\|_\infty \leqslant \|\cdot\|_p$. Donc $\|p - \widehat{p}\|_\infty \leqslant \varepsilon$, ce qui se réécrit $|p_i - \widehat{p}_i| \leqslant \varepsilon$ et ensuite $|\widehat{p}| \leqslant |p_i| + \varepsilon$ pour tout $i = 1, \ldots, n$. Ainsi

$$|z| \leqslant \max\{1, \sum_{i=1}^{n} |p_i| + n\varepsilon\} =: R.$$

On déduit de ce qui précède que

$$Z_\varepsilon(p) \subset B(0, R) \text{ boule fermée de centre 0 et de rayon } R.$$

Nous pouvons alors définir la grille par $[-R, R] \times [-R, R]$.
Le problème de cette méthode est que si les coefficients du polynôme sont grands alors la grille est très grande même si les racines sont petites. Une solution pourrait être d'utiliser d'autres bornes sur les racines, voir de choisir les bornes en fonction de la taille des coefficients.

Discutons maintenant la finesse de la grille. Si l'on veut un ensemble de pseudozéros, il faut que le choix de la grille permette d'isoler les racines de p. Le pas de la grille doit donc être choisi en conséquence. Il existe une grande quantité de résultats concernant la séparation des racines. Les bornes suivantes sur la séparation des racines sont proposées par Mahler et Rump [207]. Si $\{z_i\}$ sont les racines distinctes d'un polynôme p de degré n, alors

$$\begin{aligned} \min |z_i - z_j| &> \sqrt{\tfrac{8}{n^{n+2}}} \tfrac{1}{1+\|p\|_\infty^n} =: \gamma_1 \quad \text{(Rump)}, \\ \min |z_i - z_j| &> \sqrt{\tfrac{|\Delta|}{n^{n+2}}} \tfrac{1}{\|p\|_2^{n-1}} =: \gamma_2 \quad \text{(Mignotte)}, \end{aligned}$$

où Δ dénote le discriminant du polynôme.

Intéressons nous maintenant au problème de précision finie. Nous avons vu précédemment que le calcul de pseudozéros revenait à l'évaluation en les points d'une grille d'une certaine fonction g définie par $g(z) = p(z)/f(z)$ où p est un polynôme et f est une norme. Il se trouve que pour les normes usuelles, on a $f(z) \geqslant 1$ et que l'erreur numérique associée est négligeable. Seule l'erreur sur l'évaluation du polynôme p est à prendre en compte. On peut utiliser l'algorithme de Horner compensé du chapitre 3 pour évaluer précisemment le polynôme. Néanmoins, dans la pratique, la taille des perturbations ε est supérieure à la précision des calculs et donc l'évaluation par l'algorithme de Horner classique est généralement suffisante.

5.4 Étude des perturbations linéaires

Nous reprenons ici l'étude des pseudozéros dans le cadre de perturbations dite *linéaires*. On perturbe le polynôme de la façon suivante

$$\widehat{p}(z) = p(z) + \varepsilon q(z),$$

où q est un polynôme vérifiant $\deg q \leqslant \deg p$. On définit alors l'ensemble des pseudozéros $Z_\varepsilon(p)$ par

$$Z_\varepsilon(p) = \{z \in \mathbf{C} : \exists \eta \text{ avec } |\eta| \leqslant \varepsilon \text{ vérifiant } p(z) + \eta q(z) = 0\}.$$

On peut calculer cet ensemble grâce à la proposition suivante.

Proposition 5.1. *L'ensemble des pseudozéros pour une perturbation linéaire vérifie*

$$Z_\varepsilon(p) = \{z \in \mathbf{C} : \left|\frac{p(z)}{q(z)}\right| \leqslant \varepsilon\}.$$

Démonstration. Il s'agit juste de réordonner les termes dans la définition ci-dessus. ■

Remarque. Si $q(z) = 0$ alors deux cas sont possibles. Si z est une racine de p alors $p(z)/q(z) = 0$ et z appartient à l'ensemble $Z_\varepsilon(p)$. Si maintenant $p(z) \neq 0$ alors alors $p(z)/q(z) = \infty$ et donc z n'est pas un pseudozéro. Il n'y a donc pas de problème dans le calcul moyennant le fait de considérer les égalités suivantes : $0/0 = 0$ et $1/0 = \infty$.

5.5 Quelques théorèmes relatifs aux pseudozéros

Moyennant quelques hypothèses sur ε et sur le coefficient dominant de p (que l'on peut supprimer si on prend p sous forme unitaire) on peut affirmer que l'ensemble des pseudozéros est un ensemble borné. Nous supposerons dans la suite que ceci est réalisé.

Les théorèmes suivants sont dûs à Mosier [143]. Ils établissent certaines relations entre les racines du polynôme p et les racines des polynômes "proches" de lui (*i.e.*, qui appartiennent à $N_\varepsilon(p)$).

Théorème 5.3. *En se plaçant dans les hypothèses précédentes (*i.e. *l'ensemble des pseudozéros est borné), si $q \in N_\varepsilon(p)$ alors q et p ont le même nombre de racines (en comptant les multiplicités) dans chaque composante connexe de $Z_\varepsilon(p)$. De plus, il y a au moins une racine du polynôme p dans chaque composante connexe de $Z_\varepsilon(p)$.*

Démonstration. Nous retranscrivons la preuve se trouvant dans [143, p. 269].
Soit $z \in Z_\mu$ ou Z_μ est une composante connexe de $Z_\varepsilon(p)$. Par définition, il existe $\widehat{p} \in N_\varepsilon(p)$ tel que $\widehat{p}(z) = 0$. Notons maintenant m_μ le nombre de zéros de p se situant dans Z_μ (en comptant les multiplicités). Notons aussi $p_t(z) := (1-t)p(z) + t\widehat{p}(z)$ pour $t \in [0,1]$. On vérifie alors que

$$\|p_t - p\| = t\|p - \widehat{p}\| \leqslant t\varepsilon \leqslant \varepsilon \quad \text{car} \quad t \in [0,1].$$

Par conséquent $p_t \in N_\varepsilon(p)$ pour tout $t \in [0,1]$. Or les coefficients de p_t sont des fonctions de t et il est bien connu que les racines d'un polynôme sont des fonctions continues de leurs coefficients. Ainsi comme t varie de 0 à 1, les racines de p_t forment des chemins continus des racines de p_0 à celles de p_1. En effet, comme les composantes connexes de $Z_\varepsilon(p)$ sont bornées et disjointes, les zéros ne peuvent "sauter" dans une autre composante connexe ou bien partir à l'infini. Les racines restent donc dans Z_μ. Donc $\widehat{p} = p_1$ admet lui aussi m_μ racines. ■

Le théorème suivant permet de donner une condition nécessaire et suffisante pour savoir si le voisinage d'une racine contient deux racines de p.

Théorème 5.4. *Un voisinage d'une racine de p contient deux racines de p si et seulement si il contient un nombre complexe u comme racine double de p_u (défini en (5.5) dans la preuve du théorème 5.1).*

Démonstration. La preuve se trouve dans [143, p. 271]. ■

Nous allons maintenant introduire une notion de multiplicité pour une composante connexe de $Z_\varepsilon(p)$ qui nous permettra par la suite de simplifier des définitions et des énoncés de théorèmes.

Définition 5.1. La *multiplicité* d'une composante connexe Z_μ de $Z_\varepsilon(p)$ est le nombre commun de zéros dans Z_μ de tous les polynômes de $N_\varepsilon(p)$.

Il s'agit d'une tranposition de la notion de racines multiples au cas des polynômes connus avec une incertitude.

5.6 Pseudozéros dans le cas de polynômes réels

Dans le cas ou le polynôme p est réel, *i.e.* $p \in \mathbf{R}_n[x]$, les polynômes perturbés sont eux aussi réels. On définit alors le voisinage de p par $N_\varepsilon^R(p) := \{q \in \mathbf{R}[x] : \|p - q\| \leqslant \varepsilon\}$. Deux cas sont alors envisageables. Ou bien on cherche les pseudozéros réels, auxquels cas c'est quasiment identique à l'étude précédente, ou alors on cherche tous les pseudozéros complexes non réels.

Nous allons nous attacher au cas complexe. On définit alors l'ensemble des pseudozéros par $Z_\varepsilon(p) := \{z \in \mathbf{C} : \widehat{p}(z) = 0 \text{ pour } \widehat{p} \in N_\varepsilon^R(p)\}$.

Il est clair que $Z_\varepsilon(p)$ est symétrique par rapport à l'axe des réels. En effet si $z \in Z_\varepsilon(p)$ alors $\overline{z} \in Z_\varepsilon(p)$ car si $\widehat{p}(z) = 0$ et $\widehat{p} \in \mathbf{R}_n[x]$ alors $\widehat{p}(\overline{z}) = \overline{\widehat{p}(z)} = 0$. On peut alors démontrer le théorème suivant.

Théorème 5.5. *Soit Z_μ une composante connexe de $Z_\varepsilon(p)$ de multiplicité 1. Alors ou bien $Z_\mu \subset \mathbf{R}$ ou bien $Z_\mu \cap \mathbf{R} = \emptyset$.*

Démonstration. Si Z_μ contient des réels et des complexes alors Z_μ doit être symétrique par rapport à l'axe des réels. Soit $z \in \mathbf{C}\backslash\mathbf{R}$ tel que $z \in Z_\mu$. Il existe donc $\widehat{p} \in N_\varepsilon^R(p) \subset \mathbf{R}_n[x]$ tel que $\widehat{p}(z) = 0$. Par conséquent $\widehat{p}(\overline{z}) = 0$ et donc $\widehat{p}$ admet au moins deux racines dans Z_μ ce qui contredit le fait que Z_μ soit de multiplicité 1. ■

Corollaire 5.1. *Une composante connexe Z_μ de $Z_\varepsilon(p)$ vérifiant $\emptyset \neq Z_\mu \cap \mathbf{R} \neq Z_\mu$ est de multiplicité au moins 2.*

Nous avons donc quelques résultats vis à vis des perturbations réelles. Dans le cas des perturbations linéaires (section 5.4), le résultat obtenu dans le cas de perturbations complexes s'applique aussi pour les perturbations réelles.

5.7 Pseudozéros simultanés

Nous reprenons ici la terminologie de Stetter [185]. Soient $\{z_k\}_{k=1,\dots,m}$ un ensemble de pseudozéros appartenant à des composantes connexes différentes. On peut se demander s'il existe un polynôme perturbé $\widehat{p}$ appartenant à $Z_\varepsilon(p)$ vérifiant $\widehat{p}(z_k) = 0$ pour $k = 1, \dots, m$.

Définition 5.2. Un ensemble $\{z_k\}$, $k = 1, \dots, m$ est un ensemble de *ε-pseudozéros simultanés* si il existe $\widehat{p} \in N_\varepsilon(p)$ tel que $\widehat{p}(z_k) = 0$ pour $k = 1, \dots, m$.

Soit $\mathcal{E}$ la variété en $\alpha = (\alpha_j)$ définie par

$$\sum_{j=0}^{n} \alpha_j z_k^j + p(z_k) = 0 \quad \text{pour} \quad k = 1, \dots, m. \tag{5.7}$$

Le théorème suivant énonce une condition nécessaire et suffisante pour qu'un ensemble $\{z_k\}_{k=1,\dots,m}$ soit un ensemble de *ε-pseudozéros simultanés*.

Théorème 5.6. *Un ensemble $\{z_k\}_{k=1,\dots,m}$ est un ensemble de* ε-pseudozéros simultanés *si et seulement si*

$$\delta := \min_{\alpha \in \mathcal{E}} \|\alpha\| \leqslant \varepsilon. \tag{5.8}$$

Démonstration. Montrons tout d'abord la condition suffisante. Soit $\beta = (\beta_j)_{j=1,\dots,n} \in \mathcal{E}$ vérifiant $\|\beta\| \leqslant \varepsilon$. On remarque alors que

$$\widehat{p}(z) = p(z) + \sum_{j=0}^{n} \beta_j z^j \in N_\varepsilon(p).$$

En effet,

$$\|\widehat{p} - p\| = \|\sum_{j=0}^{n} \beta_j z^j\| = \|\beta\| \leqslant \varepsilon.$$

Et comme $\beta \in \mathcal{E}$, on a : $\widehat{p}(z_k) = 0$ pour $k = 1, \dots, m$.

Pour la condition nécessaire, on suppose l'existence de $\widehat{p} \in N_\varepsilon(p)$ vérifiant $\widehat{p}(z_k) = 0$ pour $k = 1, \dots, m$.
Définissons $f(z) = \widehat{p}(z) - p(z) = \sum_{i=0}^{n} f_i z^i$. Comme $\widehat{p} \in N_\varepsilon(p)$, on a $\|f\| \leqslant \varepsilon$. De plus $\widehat{p}(z_k) = 0$ pour $k = 1, \dots, m$. Par conséquent,

$$\sum_{j=0}^{n} f_j z^j + p(z_k) = 0 \quad \text{pour} \quad k = 1, \dots, m.$$

Donc $f = (f_j) \in \mathcal{E}$ et $\delta \leqslant \varepsilon$. ■

Remarque. Dans le cas particulier ou $n = m$, l'équation (5.7) admet une solution unique (sous réserve bien sûr que la matrice associée au système soit inversible). La variété étant réduite à un unique point, il suffit de calculer la distance de ce point à l'origine et de la comparer à ε.

On peut aussi définir des ensembles de pseudozéros simultanés en restreignant les ensembles de pseudozéros. Supposons par exemple avoir fixé un pseudozéro z_1 d'un polynôme p. On peut restreindre l'ensemble des pseudozéros en disant qu'il s'agit des racines des polynômes perturbés $\widehat{p}$ vérifiant $\widehat{p}(z_1) = 0$.

Définition 5.3. Pour $z_1 \in Z_\varepsilon(p)$, on définit l'ensemble

$$Z_\varepsilon(p \mid z_1) := \{z \in \mathbf{C} : \exists \widehat{p} \in N_\varepsilon(p) \text{ avec } \widehat{p}(z_1) = 0 \text{ et } \widehat{p}(z) = 0\}.$$

On peut alors énoncer le théorème suivant, dont la preuve est immédiate.

Théorème 5.7. *$z \in Z_\varepsilon(p \mid z_1)$ si et seulement si z et z_1 sont des pseudozéros simultanés de p.*

5.8 Pseudozéros avec multiplicité

Définition 5.4. Un pseudozéro z de p est dit de *multiplicité m* si il existe un polynôme perturbé $\widehat{p}$ appartenant à $N_\varepsilon(p)$ ayant z comme racine avec la multiplicité m.

On sait que z est une racine de multiplicité m de $\widehat{p}$ si et seulement si

$$\widehat{p}^{(k)}(z) = 0 \quad \text{pour} \quad k = 0, \ldots, m-1. \tag{5.9}$$

En notant que $\widehat{p}(x) = p(x) + \sum_{j=0}^{n} \alpha_j x^j$ et en utilisant (5.9), on obtient

$$\sum_{j=0}^{n-k} \binom{j}{k} \alpha_j z^{j-k} + \frac{1}{k!} p^{(k)}(z) = 0, \quad k = 0, \ldots, m-1. \tag{5.10}$$

L'équation (5.10) est l'équation d'une variété linéaire de $\mathbf{C}^n$ que nous noterons $\mathcal{E}^{(m)}(z)$. Nous avons alors le corollaire suivant.

Corollaire 5.2. *$z \in \mathbf{C}$ est un ε-pseudozéro de multiplicité m si et seulement si $\mathcal{E}^{(m)}(z)$ vérifie (5.8).*

5.9 Simulations numériques

Dans cette section, nous proposons quelques simulations numériques obtenues avec le package que nous avons écrit en MATLAB.

La figure 5.2 représente l'ensemble des pseudozéros du polynôme $p(z) = z^2 - (10.5 + 10.2i)z + (1.5 + i53.5)$ pour différentes valeurs de ε. Les perturbations du polynômes sont de type *par composante* pondérées par $f_0 = 4$, $f_1 = 0.5$, $f_2 = 0.01$. On peut remarquer qu'avec une tolérance de 0.01, on ne peut distinguer les deux racines de p qui sont $5 + i5$ et $5.5 + i5.2$. En diminuant la tolérance, on voit alors la séparation des deux racines.

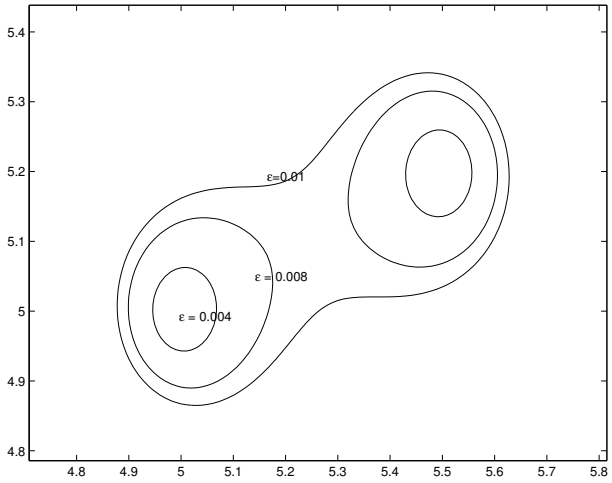


FIG. 5.2 – Ensembles des pseudozéros du polynôme $p(z) = z^2-(10.5+10.2i)z+(1.5+i53.5)$ pour trois valeurs différentes de ε.

La figure 5.3 montre l'évolution de l'ensemble des pseudozéros $Z_{\varepsilon\|p\|_d}(p)$ du polynôme $p(z) = 1 + z + \cdots + z^{20}$ dont les racines sont les racines 21^{e} de l'unité. Nous utilisons la norme $\|\cdot\|_d$ de Trefethen et Toh [192] (voir la remarque page 71) avec $d = \|p\|_2 p^{-1}$.

La figure 6.2 montre l'évolution de l'ensemble des pseudozéros $Z_{\varepsilon\|p\|_d}(p)$ du polynôme p ayant pour racines $2^{-10}, 2^{-9}, \ldots, 2^9$ avec la même norme que précédemment. On voit très clairement la variation d'incertitude sur les racines en fonction de la tolérance ε.

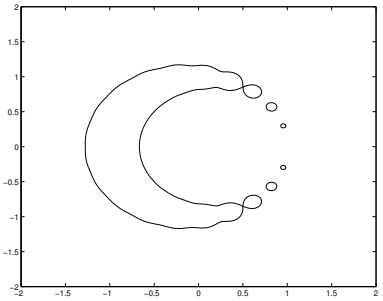
La figure 5.5 montre d'une part les valeurs de $g(z)$ (voir remarque page 73) sur la grille que nous avons choisi et d'autre part l'ensemble des pseudozéros (en norme 2) $Z_\varepsilon(p)$ pour $\varepsilon = 0.1$ du polynôme $p(z) = (z-1)(z-2)$.

5.10 Conclusion sur les pseudozéros

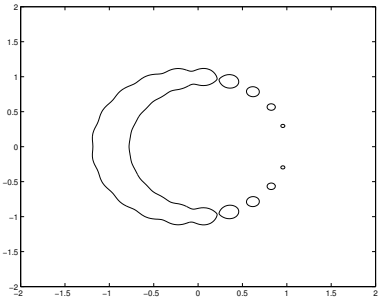
Les pseudozéros semblent être un outil de choix dans l'étude des racines des polynômes en précision finie. Ils aident à la compréhension des problèmes numériques liés à la recherche de racines. C'est un outil complémentaire au conditionnement qui permet de plus une vision géométrique du problème.

Mais l'intérêt principal des pseudozéros est qu'ils permettent de transposer les perturbations sur les coefficients des polynômes en des perturbations sur les racines.

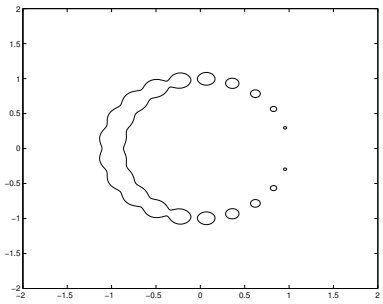
Dans les chapitres suivants, nous allons proposer d'autres applications dans différents domaines. Dans le chapitre 6, nous donnons des applications en calcul formel. Dans le chapitre 7, nous proposons des applications en théorie du contrôle.



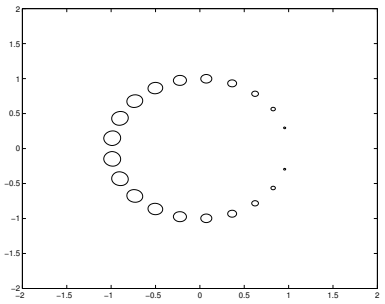
(a) $\varepsilon = 10^{-1}$



(b) $\varepsilon = 10^{-1.1}$

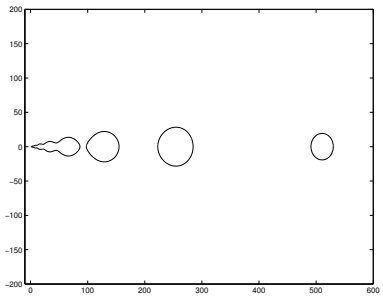


(c) $\varepsilon = 10^{-1.2}$

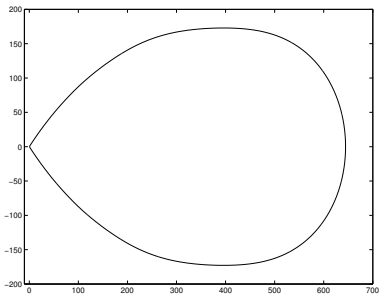


(d) $\varepsilon = 10^{-1.35}$

FIG. 5.3 – Ensemble des pseudozéros pour diverses valeurs de ε du polynôme $p(z) = 1 + z + \cdots + z^{20}$.

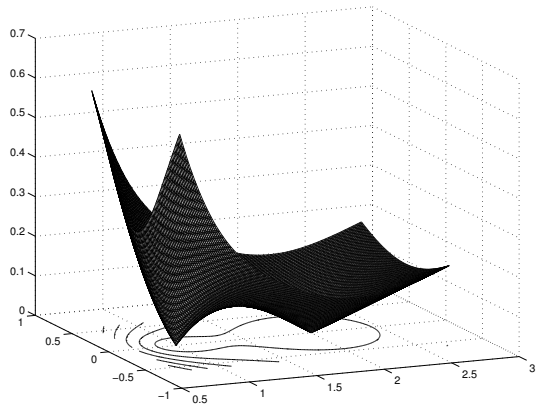


(a) $\varepsilon = 10^{-3}$

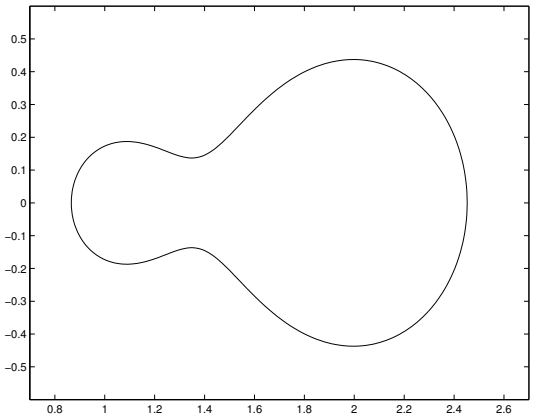


(b) $\varepsilon = 10^{-2}$

FIG. 5.4 – Ensemble des pseudozéros pour diverses valeurs de ε du polynôme p ayant pour racine $2^{-10}, 2^{-9}, \ldots, 2^{9}$.



(a) valeurs de $g(z)$ aux points de la grille



(b) coupe horizontale dans la figure (a)

FIG. 5.5 – Ensemble des pseudozéros pour $\varepsilon = 0.1$ du polynôme $p(z) = (z-1)(z-2)$.

Chapitre 6

Applications des pseudozéros en calcul formel

6.1 Introduction

On va montrer dans cette introduction la limite de la définition du PGCD algébrique dans le domaine de la précision finie. Prenons par exemple deux polynômes p et q unitaires et tels que $\deg p > 1$. On suppose de plus que p divise q. Cela revient à dire que $\mathrm{PGCD}(p, q) = p$. Or pour toute constante $\varepsilon > 0$, on a $\mathrm{PGCD}(p, q + \varepsilon) = 1$. Par conséquent, une petite perturbation du polynôme p peut entraîner un « saut » important du PGCD. Le PGCD ne dépend donc pas continûment des perturbations de ses coefficients. Le calcul du PGCD est par conséquent un problème mal posé au sens d'Hadamard.

On a le même type de problèmes avec la notion de polynômes « premiers entre eux ». L'exemple suivant issu de [12] est révélateur. Soient p et q les polynômes suivant

$$p(z) = (z - \frac{1}{3})(z - \frac{5}{3}) = z^2 - 2z + \frac{5}{9}, \quad q(z) = z - \frac{1}{3}.$$

Lorsque l'on transforme les coefficients de p et q en nombres flottants, les deux polynômes, qui ont une racine commune en arithmétique exacte, deviennent premiers entre eux. De la même façon, les polynômes

$$p(z) = 50z - 7, \quad q(z) = z - \frac{1}{7},$$

premiers entre eux en arithmétique exacte, ne le sont plus en considérant une précision de deux chiffres après la virgule ($1/7 = 0.14285714$ et $7/50 = 0.14$).

L'idée dans l'aspect approché est de ne plus considérer les polynômes comme des données exactes mais plutôt comme des boules dans l'espace des polynômes. On s'attend alors à ce que, après une légère perturbation des polynômes, on obtienne un PGCD approché proche du PGCD exact.

6.2 PGCD approché de polynômes

6.2.1 Historique

Le calcul du PGCD approché a été relancé en 1985 par A. Schönhage [178] sous le terme de « Quasi-GCD ». Il suppose dans son étude qu'il peut avoir une approximation arbitrairement précise des réels. Autrement dit il travaille en précision infinie.

Pour $f \in \mathcal{P}_n$, on notera $\rho(f)$ le réel défini par $\rho(f) = \max_{z \text{ racine de } f} |z|$. Le problème qu'il résout est alors le suivant :

> Étant donnés $\|\cdot\|_1$ la norme 1, $f \in \mathcal{P}_n$, $g \in \mathcal{P}_m$ avec $1 \leqslant m < n$, $\|f\|_1$, $\|g\|_1 \in [\frac{1}{2}, 1]$, $\rho(f) \leqslant \frac{1}{4}$ et étant donné $0 < \varepsilon \leqslant \frac{1}{2}$, trouver des polynômes h ,$u \in \mathcal{P}_{m-1}$, $v \in \mathcal{P}_{n-1}$ tels que
>
> (i) $\|hf_1 - f\|_1 < \varepsilon$, $\|hg_1 - g\|_1 < \varepsilon$ pour f_1, g_1 bien choisis
>
> (ii) $\|uf + vg - h\|_1 < \varepsilon \|h\|_1$

Le polynôme h est appelé un *Quasi-GCD* de p et q. Il résout ce problème en utilisant un algorithme d'Euclide modifié de façon à le rendre plus stable.

Mais le problème est que généralement, on ne peut avoir une précision infinie. En effet, dès que l'on travaille sur une machine, les réels sont approximés en nombres flottants et par conséquent les données que l'on code sur la machine sont entachées d'erreurs. La définition de PGCD approché a du évoluer. C'est ce que nous allons étudier dans les sections suivantes.

6.2.2 Définition du ε-PGCD

Il y a plusieurs façon de définir un ε-PGCD mais la plus couramment admise est la suivante.

Définition 6.1. Étant donnés deux polynômes p et q de degré respectif n et m, et ε un réel strictement positif, on appelle *ε-diviseur* (ou *diviseur approché*) de p et q tout diviseur des polynômes perturbés $\widehat{p}$ et $\widehat{q}$ vérifiant $\|p - \widehat{p}\| \leqslant \varepsilon$, $\|q - \widehat{q}\| \leqslant \varepsilon$ et $\deg(p - \widehat{p}) \leqslant n$, $\deg(q - \widehat{q}) \leqslant m$. Un *$\varepsilon$-PGCD* de p et q est un ε-diviseur de degré maximum.

Remarque. Autrement dit, ε-PGCD$(p, q) = \mathrm{PGCD}(\widehat{p}, \widehat{q})$ pour $\widehat{p} \in \mathcal{P}_n$ et $\widehat{q} \in \mathcal{P}_m$ vérifiant $\|p - \widehat{p}\| \leqslant \varepsilon$, $\|q - \widehat{q}\| \leqslant \varepsilon$ et tel que $\deg \mathrm{PGCD}(\widehat{p}, \widehat{q})$ soit maximal.
On remarquera aussi que la définition n'implique pas l'unicité du ε-PGCD. Il est clair que le degré du PGCD est unique mais en aucune manière le ε-PGCD.

6.2.3 Calcul d'un ε-PGCD

Cinq types d'algorithmes ont été proposés pour calculer un ε-PGCD.

1. Un point de vue est d'utiliser une définition plus forte pour le PGCD approché en se basant sur une représentation par les racines. L'idée est de considérer les racines des polynômes de départ et d'associer le plus possible de racines proches entre ces polynômes. Cette étude est détaillée dans [159, 161].

2. Une approche possible consiste à adapter l'algorithme classique d'Euclide pour le calcul du PGCD de deux polynômes en modifiant les tests et les conditions d'arrêt [58, 101]. Cette approche semble naturelle mais on ne peut pas prouver en sortie de l'algorithme que l'on a bien un ε-PGCD (ε-diviseur de degré maximum). On ne récupère en fait qu'un ε-diviseur.
3. Une autre approche proposée est une approche matricielle [39, 57, 58]. Elle consiste à se ramener à des problèmes matriciels (en général), et utiliser des outils numériques (par exemple la SVD (Décomposition en Valeurs Singulières)). Cette méthode permet d'obtenir une borne supérieure sur le degré d'un ε-PGCD.
4. Une autre approche consiste à formuler le problème du ε-PGCD comme un problème d'optimisation [113]. Karmarkar et Lakshman [113] propose un algorithme de calcul d'un ε-PGCD ainsi que les perturbations sur les polynômes en entrée réalisant ce PGCD approché. Une partie de leur algorithme peut se faire en précision finie mais sa complexité est exponentielle en le degré du PGCD approché. Néanmoins, il s'agit de la première approche permettant d'obtenir un PGCD approché certifié (on n'a pas seulement obtenu un diviseur approché).
5. Très récemment, Zeng [203, 205] a proposé une approche numérique qui fusionne plusieurs propriétés et algorithmes à la fois symbolique et numérique. Pour deux polynômes connus avec une certaine incertitude, il identifie le degré du ε-PGCD en appliquant successivement des SVD partielles sur les matrices de Sylvester. Cette étape permet aussi de déterminer une approximation intiale du PGCD approché afin d'effectuer ensuite une méthode de Newton qui converge vers le ε-PGCD. La fiabilité des calculs provient du fait qu'il tranforme un problème mal conditionné en un problème bien conditionné que l'on définit sur une « variété péjorative » (on connaît la multiplicité des racines du PGCD approché). Cette idéé a été introduite en 1972 par Kahan [110].

6.2.4 Une étude géométrique

Nous introduisons dans cette section une nouvelle définition de ε-PGCD à l'aide des pseudozéros. Il s'agit d'une définition proche de celle de Pan [159]. Nous allons le définir sous la forme d'un algorithme. Tout d'abord on trace les pseudozéros de p et q que l'on superpose. Lorsqu'une composante connexe de l'ensemble des pseudozéros de p intersecte une composante connexe de l'ensemble des pseudozéros de q alors on choisit un point dans l'intersection. On fait cela pour toutes les intersections entre les différentes composantes connexes. On obtient alors une suite de nombre $(\alpha_i)_{i=1,\dots,l}$. On définit alors un ε-PGCD par $\Pi_{i=1}^{l}(z-\alpha_i)$.

Certes, cet algorithme ne nous permet pas de certifier que l'on a un PGCD de degré maximal car on ne tient pas compte des multiplicités des racines. Il s'agit simplement de regrouper les racines « presque communes ».

Il apparaît que les pseudozéros ne permettent pas de résoudre le problème du PGCD approché. Nous allons montré maintenant que cet outil permet néanmoins de répondre au problème de la primalité approchée.

6.3 Polynômes premiers entre eux

Dans cette section, nous étudions la notion de polynômes *premiers entre eux*. On dit que deux polynômes sont *premiers entre eux* s'ils n'ont aucunes racines communes. Par analogie, nous dirons que deux polynômes sont ε-premiers entre eux si leur ε-PGCD est 1. Il est clair qu'un calcul de PGCD approché permet de tester la primalité approchée. Néanmoins, d'autres méthodes semblent plus efficaces. Il est souvent intéressant, avant de commencer un calcul de ε-PGCD coûteux, de faire un test de primalité approchée.

Nous étudions tout d'abord les articles [11, 12] qui donnent des bornes sur les perturbations afin que deux polynômes restent premiers entre eux. Ensuite, nous montrons que les tracés de pseudozéros permettent de tester graphiquement la primalité approchée de deux polynômes.

6.3.1 Borne sur les perturbations

6.3.1.1 Définitions et notations

Soient p et q deux polynômes de $\mathbf{C}[z]$ définis par

$$p(z) = p_n z^n + p_{n-} z + \cdots + p_0, \quad q(z) = q_m z^m + q_{m-1} z^{m-1} + \cdots + q_0, \quad p_n, q_m \neq 0.$$

Le PGCD de p et q est donné par

$$\mathrm{PGCD}(p,q)(z) = \prod_{\gamma \in A \cap B} (z-\gamma), \text{ où } p(z) = p_n \cdot \prod_{\alpha \in A}(z-\alpha), \quad q(z) = q_m \cdot \prod_{\beta \in B}(z-\beta).$$

On définit de plus sur $\mathbf{C}[z]$ la norme $\|\cdot\|$ par

$$\|p\| = \sum_j |p_j|.$$

Par extension, on définit

$$\|(p,q)\| = \max\{\|p\|, \|q\|\} = \max\{\sum |p_i|, \sum |q_j|\}.$$

On pose alors la définition suivante :

Définition 6.2. Pour $p, q \in \mathbf{C}[z]$, on définit

$$\epsilon(p,q) = \inf\{\|(p-\widehat{p}, q-\widehat{q})\| : (\widehat{p}, \widehat{q}) \text{ ont une racine commune et } \deg\widehat{p} \leqslant n, \deg\widehat{q} \leqslant m\},$$

Remarque. Ceci signifie que tous polynômes $\widetilde{p}, \widetilde{q}$ vérifiant $\|(p-\widetilde{p}, q-\widetilde{q})\| \leqslant \varepsilon < \epsilon(p,q)$ avec $\deg\widetilde{p} \leqslant n, \deg\widetilde{q} \leqslant m$ sont premiers entre eux. Les polynômes p et q sont dit ε-premiers.

6.3.1.2 Calcul d'une borne inférieure pour $\epsilon(p,q)$

Il est bien connu que le calcul de PGCD peut être vu comme un problème d'algèbre linéaire. Soit $S(p,q)$ la matrice de Sylvester de (p,q),

$$S(p,q) = \begin{bmatrix} p_0 & 0 & \cdots & 0 & q_0 & 0 & \cdots & 0 \\ p_1 & p_0 & \ddots & \vdots & q_1 & q_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots & \ddots & \ddots & 0 \\ p_n & & \ddots & p_0 & q_m & & \ddots & q_0 \\ 0 & p_n & & p_1 & 0 & q_m & & q_1 \\ \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & p_n & 0 & \dots & 0 & q_m \end{bmatrix} \in \mathbf{C}^{(n+m)\times(n+m)}.$$

Le critère de Sylvester affirme que deux polynômes sont *premiers entre eux* si et seulement si $S(p,q)$ est une matrice *inversible*. Berkermann et Labahn ont montré [12] que

Lemme 6.1. *Pour tout polynôme p et q, nous avons*

$$\epsilon(p,q) \geqslant \frac{1}{\|S(p,q)^{-1}\|}. \tag{6.1}$$

Démonstration. Elle repose sur le théorème de Gastinel (voir [12]). ■

Dans [12], Berkermann et Labahn proposent de calculer une borne supérieure de $\|S(p,q)^{-1}\|$.

6.3.1.3 Calcul d'une borne inférieure pour $\|S(p,q)^{-1}\|$

Le théorème de Bezout assure que p et q sont premiers entre eux si et seulement si il existe $u, v \in \mathbf{C}[z]$, avec $\deg u < n, \deg v < m$, satisfaisant

$$p \cdot u + q \cdot v = 1. \tag{6.2}$$

L'équation (6.2) peut se réécrire matriciellement sous la forme

$$S(p,q) \cdot \begin{bmatrix} \vec{v} \\ \vec{u} \end{bmatrix} = (1, 0, \dots, 0)^T. \tag{6.3}$$

Ainsi, deux polynômes sont premiers entre eux si et seulement si on peut déterminer la première colonne de l'inverse de la matrice de Sylvester correspondante.

En explicitant l'inverse de la matrice de Sylvester, on peut montrer le

Théorème 6.1. *Soient u et v deux polynômes de degrés au plus $n-1$ et $m-1$ satisfaisant (6.2). Alors*

$$\left\| \begin{bmatrix} v \\ u \end{bmatrix} \right\| \leqslant \|S(p,q)^{-1}\| \leqslant \left\| \begin{bmatrix} v \\ u \end{bmatrix} \right\| + 2 \cdot \|f\| \cdot \|(p,q)\| \tag{6.4}$$

ou f est donnée par $f(z) = f_{-1}z^{-1} + \ldots + f_{1-n-m}z^{1-n-m} = \frac{u(z)}{p(z)} + \mathcal{O}(z^{-n-m})_{z\to\infty}$.

Démonstration. Voir [12]. ■

Introduisons le problème dual de (6.2) qui est équivalent à l'existence de $\underline{u}, \underline{v} \in \mathbf{C}[z]$ avec $\deg \underline{u} < m, \deg \underline{v} < n$ vérifiant

$$p(z) \cdot \underline{v}(z) + q(z) \cdot \underline{u}(z) = z^{n+m+1}, \tag{6.5}$$

ce qui se réécrit matriciellement sous la forme

$$S(p,q) \begin{bmatrix} \vec{v} \\ \vec{\underline{u}} \end{bmatrix} = (0, \ldots, 0, 1)^T. \tag{6.6}$$

On peut alors montrer que

$$f(z) = z^{1-n-m}[\underline{v}(z) \cdot u(z) - \underline{u}(z) \cdot v(z)]. \tag{6.7}$$

En notant

$$\kappa = \left\| \begin{bmatrix} v & \underline{v} \\ u & \underline{u} \end{bmatrix} \right\| = \max\{ \left\| \begin{bmatrix} v \\ u \end{bmatrix} \right\|, \left\| \begin{bmatrix} \underline{v} \\ \underline{u} \end{bmatrix} \right\| \}$$

et en combinant le théorème 6.1 et la relation (6.6), on obtient le

Théorème 6.2. *Soient u, v et $\underline{u}, \underline{v}$ solution de (6.2) et (6.6). Nous avons*

$$\kappa \leqslant \|S(p,q)^{-1}\| \leqslant \kappa + 2 \cdot \|f\| \cdot \|(p,q)\|,$$

avec $\|f\| = \|\underline{v} \cdot u - \underline{u} \cdot v\|$. De plus, on a la majoration suivante, $\|f\| \leqslant \kappa^2$.

En reprenant l'inégalité (6.1) et le théorème précédent, on obtient une borne inférieure de $\epsilon(p,q)$,

$$\epsilon(p,q) \geqslant \frac{1}{\kappa + 2\kappa^2 \cdot \|(p,q)\|}. \tag{6.8}$$

Or numériquement, on trouve que $\|S(p,q)^{-1}\|$ est proportionnel à κ et non à κ^2. Il semble donc que la borne trouvée ne soit pas optimale. Afin d'obtenir une borne plus précise, nous devons effectuer une étude plus détaillée de $\epsilon(p,q)$.

On peut montrer le résultat suivant :

Théorème 6.3. *Nous avons*

$$\epsilon(p,q) = \inf_{z \in \overline{\mathbf{C}}} \left\| \frac{p(z)}{\|(1,z^n)\|}, \frac{q(z)}{\|(1,z^m)\|} \right\| \tag{6.9}$$

où $\overline{\mathbf{C}} = \mathbf{C} \cup \{\infty\}$.

Si on note $h(p,q,z) = \inf_{z \in \overline{\mathbf{C}}} \| \frac{p(z)}{\|(1,z^n)\|}, \frac{q(z)}{\|(1,z^m)\|} \|$ et z^* le point où h atteint son minimum, on montre alors que z^* est la racine commune des polynômes perturbés $\widehat{p}$ et $\widehat{q}$. Il résulte de cela que si z^* appartient au disque unité alors

$$\epsilon(p,q) = \inf_{|z| \leqslant 1} \|(p(z), q(z))\| \geqslant \frac{1}{\|(v,u)^T\|}$$

sinon

$$\epsilon(p,q) = \inf_{|z|\geqslant 1} \|(p(z), q(z))\| \geqslant \frac{1}{\|(\underline{v}, \underline{u})^T\|}.$$

Finalement, en regroupant les deux inégalités précédentes, on obtient

$$\epsilon(p,q) \geqslant \frac{1}{\kappa}. \tag{6.10}$$

Cet algorithme a été intrégré dans la version 8 de Maple dans le package SNAP [108]. Cet algorithme nécessite $\mathcal{O}((n+m)^2)$ opérations mais ne donne pas toujours une borne précise pour $\epsilon(p,q)$ comme nous allons le montrer.

Prenons par exemple les polynômes $p(z) = z^2$ et $q(z) = (z-1)^2$. La fonction `DistanceToCommonDivisors` du package SNAP donne 0.125 comme borne inférieure de $\epsilon(p,q)$. Maintenant, en utilisant la fonction `AreCoprime` par la commande

```
> AreCoprime(p,q,z,0.2);
```

on obtient `FAIL`. Cela signifie que le logiciel n'est pas capable de décider de la 0.2-primalité de p et q.

À notre connaissance, il n'existe pas d'autres algorithmes pour décider l'ε-primalité de polynômes sans avoir à calculer un ε-PGCD. Nous allons montrer dans la sous-section 6.3.3) que l'on peut utiliser le tracé de pseudozéros pour tester la primalité approchée. Nous expliquerons comment la figure 6.1 montre que les polynômes p et q sont 0.2-premiers.

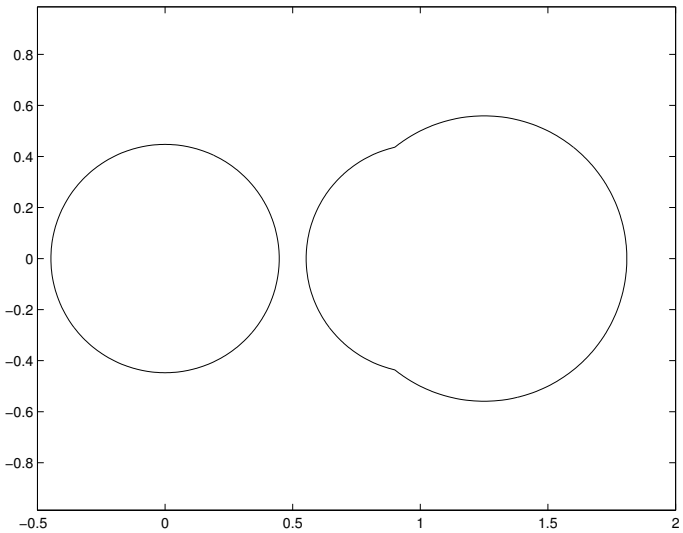


FIG. 6.1 – L'ensemble des ε-pseudozéros montre que les polynômes $p = z^2$ et $q(z) = (z-1)^2$ sont 0.2-premiers ($\varepsilon = 0.2$)

6.3.2 Utilisation d'une SVD

Nous avons dit, dans le chapitre précédent, que l'utilisation d'une SVD permet d'avoir une borne supérieure du degré d'un ε-PGCD. Par conséquent, si l'on trouve que le degré d'un ε-PGCD est 0 alors on est sûr que les deux polynômes sont premiers entre eux. Sinon, on ne peut rien dire. Le coût de cet algorithme est en $\mathcal{O}((n+m)^3)$.

6.3.3 Aspect géométrique à l'aide des pseudozéros

Nous allons maintenant étudier ce que la notion de pseudozéros peut apporter à l'étude de la primalité. De part la définition des pseudozéros, nous pouvons affirmer deux choses.

Proposition 6.1. *Soient p et q deux polynômes de $\mathcal{P}_n$. On a les deux assertions suivantes :*

(1) si l'intersection des pseudozéros est vide alors les deux polynômes sont premiers entre eux,

(2) si l'intersection est non vide alors ils ont un ε-PGCD non trivial.

Démonstration. Montrons ces affirmations :

Soient p et q deux polynômes à coefficients complexes. Si $Z_\varepsilon(p) \cap Z_\varepsilon(q) = \emptyset$ alors par définition de l'ensemble des pseudozéros, on ne peut pas trouver $\widehat{p} \in N_\varepsilon(p)$ et $\widehat{q} \in N_\varepsilon(q)$ ayant une racine commune. Cela signifie bien que p et q sont ε-premier entre eux. Si maintenant $Z_\varepsilon(p) \cap Z_\varepsilon(q) \neq \emptyset$ alors prenons $a \in Z_\varepsilon(p) \cap Z_\varepsilon(q)$. Cela signifie qu'il existe $\widehat{p} \in N_\varepsilon(p)$ et $\widehat{q} \in N_\varepsilon(q)$ tels que $\widehat{p}(a) = 0$ et $\widehat{q}(a) = 0$. Donc le polynôme $(z-a)$ divise $\widehat{p}$ et $\widehat{q}$. Par conséquent un ε-PGCD est au moins de degré 1 et donc p et q ne sont pas ε-premiers entre eux. ■

Appliquons cette proposition en considérant, par exemple, p et q définis par

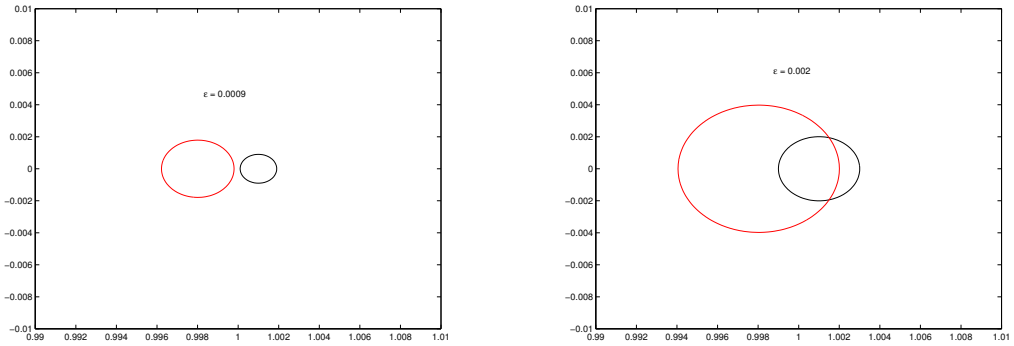
$$p(z) = z^2 - 3.999z + 3.001, \quad \text{et} \quad q(z) = z^2 - 3.001z + 1.999.$$

La figure 6.2 représente l'ensemble des ε-pseudozéros de ces deux polynômes pour deux valeurs différentes de ε (0.0009 et 0.002). Sur la figure de gauche, l'intersection est vide donc les deux polynômes p et q sont 0.0009-premiers entre eux. Par contre, sur la figure de droite, l'intersection est non vide si bien que p et q ne sont pas 0.002-premiers entre eux.

Néanmoins un problème se pose. Les composantes connexes d'un ensemble de pseudozéros sont convexes. Mais si la discrétisation de la grille n'est pas assez fine, il se peut que l'on trouve que l'intersection est vide alors qu'elle ne l'est pas en réalité, comme le montre la figure 6.3.

6.3.4 Synthèse

On récapitule ici les différentes méthodes pour tester la ε-primalité. Les deux premières méthodes du tableau 6.1 sont issues de la littérature. La troisième est notre contribution au sujet.



(a) p et q sont 0.0009-premiers entre eux (b) p et q ont un 0.002-diviseur commun

FIG. 6.2 – Ensemble des ε-pseudozéros pour différentes valeurs de ε des polynômes p et q.

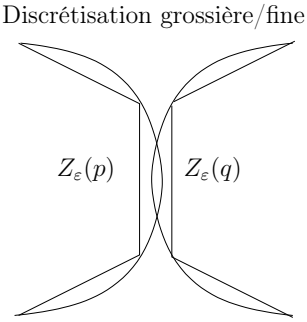


FIG. 6.3 – Influence de la discrétisation dans le choix de la primalité.

Méthode	Coût	Certification
Borne de Beckermann et Labahn	$\mathcal{O}((n+m)^2)$	primalité
SVD	$\mathcal{O}((n+m)^3)$	primalité
pseudozéros	$\times$	non primalité

TAB. 6.1 – Différentes méthodes pour tester l'ε-primalité.

Notre algorithme est avant tout un algorithme qualitatif. Il permet d'apprécier graphiquement la primalité de deux polynômes définis avec une certaine incertitude. Il est difficile d'automatiser le calcul explicite de l'intersection de deux ensembles de pseudozéros. Nous verrons au chapitre 7 que l'on peut le faire pour l'intersection entre une droite et l'ensemble des pseudozéros d'un polynôme

Chapitre 7

Applications des pseudozéros en théorie du contrôle

En théorie du contrôle et en automatique, on écrit classiquement les fonctions de transfert sous la forme $H(p) = N(p)/D(p)$, ou N et D sont deux polynômes et où p est le paramètre du système. Le système décrit par cette fonction de transfert est dit *stable* (au sens de Hurwitz) si tous les zéros de D sont à parties réelles négatives (autrement dit si le polynôme D est stable au sens de Hurwitz). Puisque des incertitudes sur les coefficients sont inévitables dans les problèmes de la vie réelle (incertitudes sur les données, erreurs d'arrondi), il est souvent utile de mesurer la distance d'un système stable au système instable le plus proche. Il s'agit en fait de mesurer la distance de D au polynôme instable le plus proche. Nous nous intéressons ici à cette distance.

En utilisant la matrice compagnon, ce problème purement polynomial peut être reformulé en un problème matriciel. Une matrice $A \in \mathbf{C}^{n\times n}$ est dite *stable* si toutes ses valeurs propres sont à parties réelles négatives. Autrement, elle est dite instable. Quand la matrice A est stable, le calcul de la distance (par rapport à la norme 2 notée $\|\cdot\|$) de A à la matrice instable la plus proche,

$$\beta(A) = \min\{\|E\| : A + E \in \mathbf{C}^{n\times n} \text{ est instable}\}.$$

a été largement étudié en algèbre linéaire numérique [90, 32, 85, 92, 80]. Hinrichsen et Kelb [90] ont introduit et étudié la notion de rayon de stabilité complexe (réel) d'une matrice de taille $n \times n$. Ils ont obtenu des caractérisations et des estimations pour le rayon de stabilité complexe. Dans [32], Byers propose un algorithme de dichotomie pour calculer ce rayon. La méthode proposée nécessite de tester si une matrice Hamiltonienne a des valeurs propres sur l'axe imaginaire, ce qui est assez coûteux. On peut trouver un état de l'art sur le problème de la matrice la plus proche ayant une propriété donnée dans [85] avec en particulier le problème de la matrice instable la plus proche. Un état de l'art sur le rayon de stabilité se trouve dans [92]. Plus récemment, He et Watson [80] ont proposé de calculer $\beta(A) = \min_{s\in\mathbf{R}} f(s)$ avec $f(s) = \sigma_{\min}(A - siI)$ où $\sigma_{\min}$ dénote la plus petite valeur singulière. Leur algorithme est basé sur un schéma d'itération inverse afin d'obtenir un point stationnaire de la fonction f. Comme pour les méthodes présentées

précédemment, ils ont besoin de vérifier si une matrice Hamiltonienne a ou non des valeurs propres sur l'axe imaginaire.

Pour s'assurer que la distance minimale est le rayon de stabilité du polynôme associé, la matrice perturbée $A+E$ doit conserver la structure de matrice compagnon de A. Cela signifie que l'on doit calculer un rayon de stabilité structuré. Dans [91, 94], les auteurs proposent différents algorithmes pour calculer cette distance avec une certaine structure. Les perturbations de la matrice A sont de la forme $A+E$ avec $E = BDC$ où $B \in \mathbf{C}^{n\times m}$ et $C \in \mathbf{C}^{p\times n}$ donnent la structure de la perturbation et $D \in \mathbf{C}^{m\times p}$ est une matrice inconnue. Soit A la matrice compagnon d'un polynôme unitaire $p = z^n + \sum_{i=0}^{n-1} p_i z^i$ de degré n,

$$A = \begin{pmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ -p_0 & -p_1 & \cdots & -p_{n-2} & -p_{n-1} \end{pmatrix}.$$

Définissons $B = [0, \ldots, 0, -1]^T \in \mathbf{C}^{n\times 1}$, $C = \mathrm{Id}_n$, et $D = [\widehat{p}_0, \ldots, \widehat{p}_{n-1}] \in \mathbf{C}^{1\times n}$. La matrice perturbée

$$A + BDC = \begin{pmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ -(p_0+\widehat{p}_0) & -(p_1+\widehat{p}_1) & \cdots & -(p_{n-2}+\widehat{p}_{n-2}) & -(p_{n-1}+\widehat{p}_{n-1}) \end{pmatrix}$$

est la matrice compagnon du polynôme perturbé $q = z^n + \sum_{i=0}^{n-1}(p_i + \widehat{p}_i)z^i$. Dans ce cas, le rayon de stabilité du polynôme perturbé p est égal à

$$\min\{\|D\| : A + BDC \in \mathbf{C}^{n\times n} \text{ est instable}\}.$$

Cette méthode nécessite de transformer un problème purement polynomial en un problème matriciel qui est plus général.

Récemment, Genin, Ştefan et Van Dooren [64] ont proposé un algorithme pour calculer le rayon de stabilité complexe pour des polynômes matriciels. Le cas polynomial est donc un cas particulier dans leur étude. Néanmoins leur algorithme nécessite le calcul des racines d'un polynôme. Ce calcul est évité dans notre algorithme et remplacé par le calcul de suites de Sturm.

Dans ce chapitre, nous proposons un algorithme de calcul du rayon de stabilité en restant dans le domaine des polynômes. Le point clé de notre algorithme est l'utilisation des pseudozéros. L'algorithme proposé ici est de type symbolique-numérique (on voit aussi le terme d'algorithme hybride). Cela signifie que l'on utilise des méthodes formelles (ici les suites de Sturm) dans des procédures numériques. Une telle approche a été initiée dans [22] pour compter le nombre de valeurs propres imaginaires pures d'une matrice

Hamiltonienne. Il semble néanmoins que de telles approches ont été très peu étudiées bien qu'elles fournissent des algorithmes efficaces et précis.

Soient $\mathcal{P}_n$ l'espace des polynômes à coefficients complexes de degré au plus n et $\mathcal{M}_n$ le sous-espace des polynômes unitaires de degré n. Soit p appartenant à $\mathcal{M}_n$ écrit sous la forme

$$p(z) = \sum_{i=0}^{n} p_i z^i, \quad p_n = 1. \tag{7.1}$$

En représentant p par le vecteur $(p_0, \ldots, p_{n-1})^T$ de ses coefficients, on identifiera la norme $\|\cdot\|$ sur $\mathcal{M}_n$ avec la norme 2 de $\mathbf{C}^n$ du vecteur correspondant.

Un ε-voisinage de p est l'ensemble des polynômes de $\mathcal{M}_n$ « assez proche » de p, c'est-à-dire,

$$N_\varepsilon(p) = \{\widehat{p} \in \mathcal{M}_n : \|p - \widehat{p}\| \leqslant \varepsilon\}. \tag{7.2}$$

Les ε-pseudozéros sont alors toutes les racines de tous les polynômes contenus dans le ε-voisinage de p. Une définition non-constructive de cet ensemble est, comme précédemment,

$$Z_\varepsilon(p) = \{z \in \mathbf{C} : \widehat{p}(z) = 0 \text{ for } \widehat{p} \in N_\varepsilon(p)\}. \tag{7.3}$$

Dans les sections 7.1 et 7.2, nous rappelons des définitions et des résultats utiles à propos, respectivement, des pseudozéros et et la stabilité des polynômes. Dans la section 7.3, nous présentons un algorithme de dichotomie pour calculer le rayon de stabilité Dans la section 7.4, nous présentons des simulations numériques de cet algorithme. Dans la section 7.5, nous présentons un algorithme de calcul du pseudoabscisse. Enfin dans les sections 7.7 et 7.8, nous donnons le code des algorithmes.

7.1 Pseudozéros et rayon de stabilité

On rappelle ici la formule pour calculer les pseudozéros.

Proposition 7.1 (Toh et Trefethen [192])**.** *L'ensemble des ε-pseudozéros de p vérifie*

$$Z_\varepsilon(p) = \left\{ z \in \mathbf{C} : g(z) := \frac{|p(z)|}{\|\underline{z}\|} \leqslant \varepsilon \right\}, \tag{7.4}$$

où $\underline{z} = (1, z, \ldots, z^{n-1})^T$.

Les polynômes que nous utilisons ici sont unitaires afin d'assurer que l'ensemble des pseudozéros est borné.

Proposition 7.2. *L'ensemble des ε-pseudozéros est un compact contenu dans la boule de rayon $\|p\| + \varepsilon$.*

Démonstration. Comme la fonction g est continue, l'ensemble $Z_\varepsilon(p) = g^{-1}([0,\varepsilon])$ est fermé. Prouvons maintenant qu'il est aussi borné. Soit $\{z_j\}_{j=1:n}$ les racines de p comptées avec leurs multiplicités et notons $r = \max_j |z_j|$. On peut montrer (voir [138, p.154]) que

$$r \leqslant \|p\|.$$

Si $z \in Z_\varepsilon(p)$ alors il existe $\widehat{p} \in \mathcal{M}_n$ vérifiant à la fois $\widehat{p}(z) = 0$ et $\|p - \widehat{p}\| \leqslant \varepsilon$. Il s'en suit que $|z| \leqslant \|\widehat{p}\|$. De plus, nous savons que $|\|\widehat{p}\| - \|p\|| \leqslant \|\widehat{p} - p\| \leqslant \varepsilon$ et donc $\|\widehat{p}\| \leqslant \|p\| + \varepsilon$. Par conséquent $|z| \leqslant \|p\| + \varepsilon$, ce qui termine la démonstration. ■

Nous introduisons maintenant la fonction $h_{p,\varepsilon} : \mathbf{R}^2 \to \mathbf{R}$ définie par

$$h_{p,\varepsilon}(x,y) = |p(x+iy)|^2 - \varepsilon^2 \sum_{j=0}^{n-1} (x^2+y^2)^j. \tag{7.5}$$

Pour un x_0 fixé, cette fonction $h_{p,\varepsilon}(x_0, y)$ est un polynôme de degré $2n$ en y. De la même façon, pour un y_0 fixé, la fonction $h_{p,\varepsilon}(x, y_0)$ est aussi un polynôme de degré $2n$ en x. D'après la proposition 7.1, l'ensemble des pseudozéros $Z_\varepsilon(p)$ satisfait

$$Z_\varepsilon(p) = \{(x,y) \in \mathbf{R}^2 : h_{p,\varepsilon}(x,y) \leqslant 0\}.$$

La proposition suivante exhibe les valeurs complexes $z = x + iy$ qui décrivent la frontière de l'ensemble des ε-pseudozéros $Z_\varepsilon(p)$.

Proposition 7.3. *Nous avons $h_{p,\varepsilon}(x,y) = 0$ si et seulement s'il existe $q \in \mathcal{M}_n$ tel que $q(x+iy) = 0$ and $\|p - q\| = \varepsilon$.*

Démonstration. C'est immédiat au vu de la proposition 7.1. ■

7.2 Abscisse, rayon de stabilité et quelques extensions

Nous présentons maintenant les notions d'abscisse, de pseudoabscisse, et de rayon de stabilité en essayant d'exhiber les relations que les lient. L'*abscisse* d'un polynôme p de $\mathcal{M}_n$ est défini par

$$a(p) = \max\{\mathrm{Re}(z) : p(z) = 0\}.$$

Ainsi, un polynôme stable satisfait $a(p) < 0$. La fonction abscisse $a : \mathcal{P}_n \to \mathbf{R}$, $a \mapsto a(p)$, est continue sur $\mathcal{M}_n$. Néanmoins a n'est pas continue sur $\mathcal{P}_n$ (voir [30]). En effet, considérons le polynôme $q_t(z) = (1 - tz)p(z)$, où p est un polynôme de degré au plus $n-1$. Il est alors clair que $q_t \to p$ quand $t \to 0$, alors que $a(q_t) = 1/t$ qui peut être arbitrairement plus grand que $a(p)$.

Pour prouver la continuité de a sur $\mathcal{M}_n$, nous allons utiliser le résultat suivant connu sous le nom de « dépendance continue des racines d'un polynôme par rapport à ses coefficients ». La démonstration se trouve dans [99, 157].

Proposition 7.4 (Ostrowski [157]). *Soit*

$$p(z) = p_0 + p_1 z + \cdots + p_{n-1} z^{n-1} + z^n$$

un polynôme unitaire à coefficients complexes. Alors, pour tout $\varepsilon > 0$, il existe $\eta > 0$ tel que pour tout polynôme

$$q(z) = q_0 + q_1 z + \cdots + q_{n-1} z^{n-1} + z^n$$

satisfaisant

$$\max_{0 \leqslant i \leqslant n} |p_i - q_i| < \eta,$$

on a

$$\min_{\sigma \in \mathfrak{S}_n} \max_{1 \leqslant j \leqslant n} |x_j - y_{\sigma(j)}| < \varepsilon,$$

où (x_j) et (y_j), $j = 1, \ldots, n$, sont respectivement les racines de p et q.

Nous pouvons maintenant prouver la continuité de a sur $\mathcal{M}_n$.

Proposition 7.5. *L'application abscisse*

$$a : \mathcal{P}_n \to \mathbf{R}$$

définie par $a(p) = \max\{\mathrm{Re}(z) : p(z) = 0\}$ est continue sur $\mathcal{M}_n$.

Démonstration. Soient p appartenant à $\mathcal{M}_n$ et $\varepsilon > 0$. D'après la proposition 7.4, il existe $\eta > 0$ tel que pour tout q dans $\mathcal{M}_n$ vérifiant

$$\max_{0 \leqslant i \leqslant n} |p_i - q_i| < \eta,$$

on ait

$$\min_{\sigma \in \mathfrak{S}_n} \max_{1 \leqslant j \leqslant n} |x_j - y_{\sigma(j)}| < \varepsilon,$$

où (x_j) et (y_j), $j = 1, \ldots, n$, sont respectivement les racines de p et q. Cela signifie qu'il existe une permutation σ telle que

$$\max_{1 \leqslant j \leqslant n} |\,\mathrm{Re}(x_j) - \mathrm{Re}(y_{\sigma(j)})| \leqslant \max_{1 \leqslant j \leqslant n} |x_j - y_{\sigma(j)}| < \varepsilon.$$

Par conséquent, nous avons

$$\begin{aligned} |a(q) - a(p)| &= |\max_{1 \leqslant j \leqslant n} \mathrm{Re}(y_j) - \max_{1 \leqslant j \leqslant n} \mathrm{Re}(x_j)| \\ &= |\max_{1 \leqslant j \leqslant n} \mathrm{Re}(y_{\sigma(j)}) - \max_{1 \leqslant j \leqslant n} \mathrm{Re}(x_j)| \\ &\leqslant \max_{1 \leqslant j \leqslant n} |\,\mathrm{Re}(y_{\sigma(j)}) - \mathrm{Re}(x_j)| \\ &\leqslant \varepsilon. \end{aligned}$$

On a donc prouvé la continuité de a sur $\mathcal{M}_n$. ∎

Une extension naturelle de la notion d'abscisse, quand le polynôme n'est connu qu'avec une certaine incertitude, est celle de *pseudoabscisse* définie par

$$a_\varepsilon(p) = \max\{\mathrm{Re}(z) : z \in Z_\varepsilon(p)\}.$$

Au lieu de s'intéresser aux parties réelles des zéros de p, on s'intéresse aux parties réelles des ε-pseudozéros de p.

Le troisième paramètre est le *rayon de stabilité*, c'est-à-dire, la distance à l'ensemble des polynômes instables. Il est défini par

$$\beta(p) = \min\{\|p - q\| : q \in \mathcal{M}_n \text{ et } a(q) \geqslant 0\}. \tag{7.6}$$

Puisque l'ensemble des polynômes instables est fermé (ceci est du à la continuité de a sur $\mathcal{M}_n$), le minimum dans (7.6) est atteint. Nous pouvons maintenant lier la notion de rayon de stabilité $\beta(p)$ à l'ensemble des pseudozéros. Le rayon de stabilité est le plus grand ε pour lequel l'ensemble des ε-pseudozéros $Z_\varepsilon(p)$ reste dans le demi-plan gauche. On vérifie que l'on a

$$a_\varepsilon(p) \geqslant 0 \Longleftrightarrow \beta(p) \leqslant \varepsilon.$$

En effet, si $a_\varepsilon(p) \geqslant 0$ alors il existe $q \in \mathcal{M}_n$ tel que $\|p - q\| \leqslant \varepsilon$ et tel que $z \in \mathbf{C}$ vérifie $\mathrm{Re}(z) \geqslant 0$ et $q(z) = 0$. Par définition de β, nous savons que $\beta(p) \leqslant \varepsilon$. Réciproquement, si $\beta(p) \leqslant \varepsilon$ alors il existe $q \in \mathcal{M}_n$ tel que $a(q) \geqslant 0$, $\|p - q\| \leqslant \varepsilon$, et tel que q ait au moins une racine z dans $\mathbf{C}$ satisfaisant $\mathrm{Re}(z) \geqslant 0$. On conclut que $a_\varepsilon(p) \geqslant 0$.

Nous pouvons maintenant énoncer le résultat suivant.

Proposition 7.6. *Si le polynôme p appartenant à $\mathcal{M}_n$ n'est pas stable, il vérifie alors $\beta(p) = 0$. Autrement, si p est stable, on a $a_{\beta(p)}(p) = 0$.*

Démonstration. La première assertion est claire. Pour la seconde, en utilisant le résultat précédent pour $\varepsilon = \beta(p)$, il vient $a_{\beta(p)}(p) \geqslant 0$. Si nous supposons que $a_{\beta(p)}(p) > 0$, alors par continuité de $a_\varepsilon(p)$ par rapport à ε, il existe $\varepsilon \in]0, \beta(p)[$ tel que $a_\varepsilon(p) \geqslant 0$. Ceci est équivalent à $\beta(p) \leqslant \varepsilon < \beta(p)$ et par conséquent cela contredit le fait que $a_{\beta(p)} > 0$. Prouvons maintenant la continuité de $a_\varepsilon(p)$ par rapport à ε. Nous avons

$$\begin{aligned} a_\varepsilon(p) &= \sup\{a(q) : \|q - p\| \leqslant \varepsilon\} = \sup\{a(p + q) : \|q\| \leqslant \varepsilon\} \\ &= \sup\{a(p + \varepsilon r) : r \in \mathcal{M}_n \text{ and } \|r\| \leqslant 1\}. \end{aligned}$$

Par conséquent, nous avons $a_\varepsilon(p) = a(p + \varepsilon r_0)$, où $r_0 \in \mathcal{M}_n$ par continuité de a sur $\mathcal{M}_n$ et par compacité de l'ensemble $\{r \in \mathcal{M}_n : \|r\| \leqslant 1\}$. ∎

À partir de maintenant, nous considérons que le polynôme p est stable. Nous prouvons tout d'abord la proposition suivante.

Proposition 7.7. *Le rayon de stabilité $\beta(p)$ satisfait*

$$\beta(p) = \min\{\|p - q\| : q \in \mathcal{M}_n \text{ et } a(q) = 0\}.$$

Démonstration. Nous allons montrer que le polynôme q^* qui réalise le minimum dans $\beta(p)$ (*i.e.*, $\|p - q^*\| = \beta(p)$) vérifie $a(q^*) = 0$. En effet, si $a(q^*) > 0$, alors définissons $p_t(z) = (1-t)p(z) + tq(z)$ pour $t \in [0,1]$. Il est clair que $p_t \in \mathcal{M}_n$, $p_0(z) = p(z)$ et $p_1(z) = q^*(z)$. Soit $\varphi : [0,1] \to \mathbf{R}$ la fonction $t \mapsto a(p_t)$. Par continuité de a sur $\mathcal{M}_n$ (voir la proposition 7.5), la fonction φ est continue. Comme $a(p) = \varphi(0) < 0$ et $a(q^*) = \varphi(1) \geqslant 0$, il existe $\bar{t} \in]0,1[$ tel que $a(p_{\bar{t}}) = 0$. De plus, on remarque facilement que $\|p_{\bar{t}} - p\| = \bar{t}\|p - q^*\| < \|p - q^*\| = \beta(p)$. Ceci contredit le fait que q^* réalise le minimum de $\beta(p)$. En conclusion, nous avons $\beta(p) = \min\{\|p - q\| : q \in \mathcal{M}_n \text{ and } a(q) = 0\}$. ■

L'algorithme que nous proposons par la suite pour calculer le rayon de stabilité d'un polynôme est basé sur le théorème suivant.

Théorème 7.1. *L'équation $h_{p,\varepsilon}(0,y) = 0$ avec $y \in \mathbf{R}$, a une solution si et seulement si $\beta(p) \leqslant \varepsilon$.*

Démonstration. Si l'équation $h_{p,\varepsilon}(0,y) = 0$ pour $y \in \mathbf{R}$, a une solution u, la proposition 7.3 implique qu'il existe un polynôme $\widehat{p}$ tel que $\widehat{p}(iu) = 0$ et $\|p - \widehat{p}\| = \varepsilon$. Par définition de $\beta(p)$, cela implique que $\beta(p) \leqslant \varepsilon$.

Si maintenant $\beta(p) \leqslant \varepsilon$, il existe un polynôme q tel que $\|q - p\| \leqslant \varepsilon$ et $a(q) \geqslant 0$. C'est pourquoi au moins une racine de q a une partie réelle positive. Définissons le polynôme $p_t(z) = (1-t)p(z) + tq(z)$ avec $t \in [0,1]$. Il est clair que p_t appartient à $\mathcal{M}_n$, $p_0(z) = p(z)$ et $p_1(z) = q(z)$. De plus, $\|p_t - p\| = t\|p - q\| = t\varepsilon \leqslant \varepsilon$ pour tout $t \in [0,1]$ si bien que $p_t \in N_\varepsilon(p)$. Soit $\varphi : [0,1] \to \mathbf{R}$ la fonction $t \mapsto a(p_t)$. Par continuité de a sur $\mathcal{M}_n$ (voir la proposition 7.5), la fonction φ est continue. Comme $a(p) = \varphi(0) < 0$ et $a(q) = \varphi(1) \geqslant 0$, il existe $\bar{t} \in [0,1]$ tel que $a(p_{\bar{t}}) = 0$. Il existe $y \in \mathbf{R}$ tel que $p_{\bar{t}}(iy) = 0$. Comme l'ensemble des ε-pseudozéros $Z_\varepsilon(p)$ est compact et contient un $iy, y \in \mathbf{R}$, nous pouvons prendre l'intersection entre la ligne verticale passant par iy et l'ensemble des ε-pseudozéros. Soit iy' une point sur le frontière de cette intersection. Il satisfait $h_{p,\varepsilon}(0,y') = 0$. Cela termine la preuve. ■

Ce théorème prouve qu'un polynôme unitaire stable perturbé par des perturbations de taille inférieure à ε (par rapport à la norme 2) reste stable tant que l'équation $h_{p,\varepsilon}(0,y) = 0$ n'admet pas de solutions réelles. Les précédents résultats de continuité prouve que l'on peut identifier la norme de la plus petite perturbation qui transforme p en un polynôme instable, c'est-à-dire le rayon de stabilité $\beta(p)$. Nous allons utiliser ces résultats pour décrire un algorithme de dichotomie sur ε afin de calculer le rayon de stabilité.

7.3 Un algorithme de dichotomie pour calculer le rayon de stabilité

Nous supposons toujours que le polynôme p est stable. Comme le polynôme z^n est instable, il est clair que $0 \leqslant \beta(p) \leqslant \|p - z^n\|$. Nous proposons d'appliquer un algorithme de dichotomie pour calculer $\beta(p)$.

Introduisons les valeurs réelles γ et δ pour représenter respectivement une borne inférieure et une borne supérieure de $\beta(p)$. Nous avons toujours $\gamma \leqslant \beta(p) \leqslant \delta$. Nous

Algorithme 7.1 Calcul du rayon de stabilité par dichotomie

Entrée : un polynôme stable p et une tolérance τ
Sortie : un réel α tel que $|\alpha - \beta(p)| \leqslant \tau$

1: $\gamma := 0, \quad \delta := \|p - z^n\|$
2: **tant que** $|\gamma - \delta| > \tau$ **faire**
3: $\quad \varepsilon := \frac{\gamma+\delta}{2}$
4: $\quad$ **si** l'équation $h_{p,\varepsilon}(0, y) = 0$ a une solution réelle **alors**
5: $\quad\quad \delta := \varepsilon$
6: $\quad$ **sinon**
7: $\quad\quad \gamma := \varepsilon$
8: $\quad$ **fin si**
9: **fin tant que**
10: **retourne** $\alpha = \frac{\gamma+\delta}{2}$

introduisons le paramètre τ pour décrire une tolérance arbitraire qui mesure la précision à laquelle le rayon de stabilité $\beta(p)$ est calculé. Ce paramètre est nécessaire pour stopper le processus de dichotomie. En effet, l'algorithme se termine lorsque $|\alpha-\beta(p)| \leqslant |\delta-\gamma| \leqslant \tau$.

L'étape difficile dans l'algorithme est de tester si le polynôme $H(y) = h_{p,\varepsilon}(0, y) = h_{2n}y^{2n} + \cdots + h_1 y + h_0$ possède une racine réelle. Nous rappelons que les coefficients de H sont réels. Nous allons résoudre ce problème en utilisant des suites de Sturm. Il est bien connu que le module maximal r de racines de H vérifie (voir [138])

$$r \leqslant \frac{\|H\|}{|h_{2n}|}.$$

Par conséquent, une possible racine réelle de H appartient nécessairement à l'intervalle $[-r, r]$. Appliquons l'algorithme d'Euclide à H et à sa dérivée H'. Soient $H_0 = H$ et $H_1 = H'$; définissons $H_{i+1} = -\operatorname{rem}(H_{i-1}, H_i)$ (rem est le reste de la division euclidienne). Notons m le plus petit entier tel que $H_{m+1} = 0$. Notons aussi $v_H(-r)$ le nombre de changements de signe dans la suite des coefficients dominants de $H_0(-r), \ldots, H_m(-r)$ et $v_H(r)$ le nombre de changements de signe dans la suite des coefficients dominants de $H_0(r), \ldots, H_m(r)$. Nous avons défini ce qui s'appelle une suite de Sturm et donc H a exactement $v_H(-r) - v_H(r)$ racines réelles distinctes. En particulier $h_{p,\varepsilon}(0, y)$ possède une racine réelle et seulement si $v_H(-r) \neq v_H(r)$. On remarque donc que les suites de Sturm permettent de faire le test de la ligne 4 de l'algorithme 7.1 sans avoir à calculer les racines de $H(y) = h_{p,\varepsilon}(0, y)$. L'algorithme de Sturm nécessite $\mathcal{O}(n^2)$ opérations. C'est un algorithme symbolique.

7.4 Simulations numériques

Nous avons implanté l'algorithme 7.1afin d'effectuer les manipulations formelles nécessaires.

Pour tracer l'ensemble des pseudozéros, nous avons utilisé MATLAB avec l'algorithme 5.1.

L'exemple le plus simple est $p(z) = z + 1$. Il est clair que le polynôme instable le plus proche de p est $q(z) = z$, et donc que le rayon de stabilité est $\beta(p) = 1$. L'algorithme 7.1 donne $\beta = 0.999996$ pour une tolérance $\tau = 0.00001$. Nous pouvons tracer l'ensemble des 0.999996-pseudozéros (voir la figure 7.1) et nous vérifions que l'ensemble des pseudozéros est bien inclus dans le demi-plan gauche et qu'il est tangent à l'axe des ordonnées. Ceci confirme l'explication intuitive de l'algorithme.

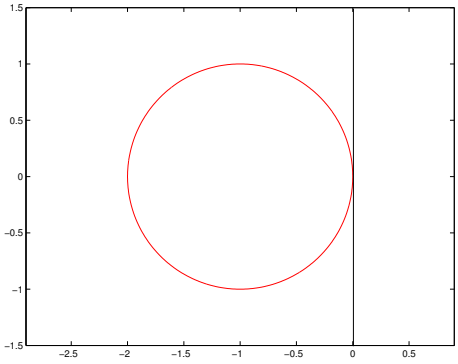


FIG. 7.1 – Ensemble des ε-pseudozéros avec $\varepsilon = 0.999996 \approx \beta(p)$ pour $p(z) = z + 1$

Si nous considérons maintenant $p(z) = (z - 1)(z - 1/2) = z^2 + z + 1/2$. Avec une tolérance $\tau = 0.00001$, nous obtenons avec l'algorithme 7.1, $\beta = 0.485868$. L'ensemble des 0.485868-pseudozéros est tracé dans la figure 7.2.

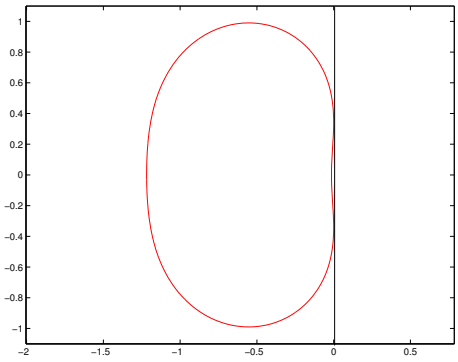


FIG. 7.2 – Ensemble des ε-pseudozéros pour $p(z) = z^2 + z + 1/2$ avec $\varepsilon = 0.485868 \approx \beta(p)$

Si nous considérons maintenant le polynôme $p(z) = z^3 + 4z^2 + 6z + 4$, nous obtenons $\beta = 2.610226$ avec une tolérance 0.00001. Nous pouvons encore tracer l'ensemble des 2.610226-pseudozéros (voir la figure 7.3).

Si nous considérons maintenant le polynôme $p(z) = z^5 + 5z^4 + 10z^3 + 10z^2 + 5z + 1$, nous obtenons $\beta = 1.000003321$ avec une tolérance 0.00001. Nous pouvons encore tracer l'ensemble des 1.000003321-pseudozéros (voir la figure 7.4).

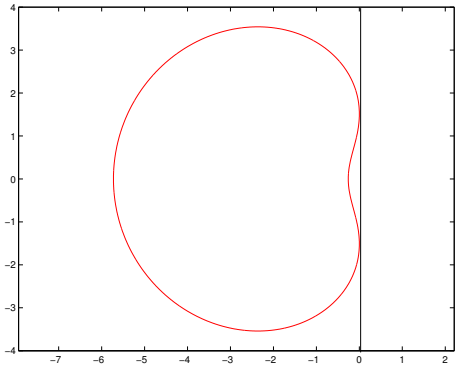


FIG. 7.3 – Ensemble des ε-pseudozéros pour $p(z) = z^3 + 4z^2 + 6z + 4$ avec $\varepsilon = 2.610226 \approx \beta(p)$

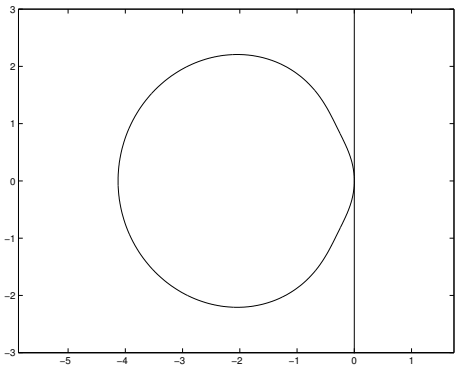


FIG. 7.4 – Ensemble des ε-pseudozéros pour $p(z) = z^5 + 5z^4 + 10z^3 + 10z^2 + 5z + 1$ avec $\varepsilon = 1.000003321 \approx \beta(p)$

Si nous considérons maintenant le polynôme $p(z) = z^6+4z^5+4z^4+6z^3+3z^2+2z+1/2$, nous obtenons $\beta = 0.08476385681$ avec une tolérance 0.00001. Nous pouvons encore tracer l'ensemble des 0.08476385681-pseudozéros (voir la figure 7.5).

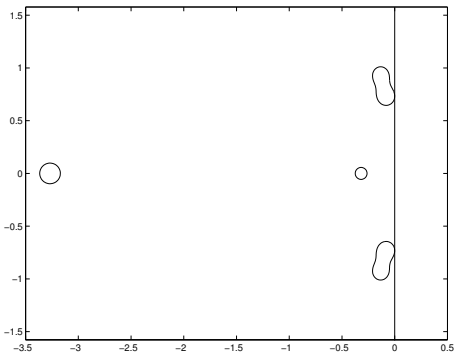


FIG. 7.5 – Ensemble des ε-pseudozéros pour $p(z) = z^6 + 4z^5 + 4z^4 + 6z^3 + 3z^2 + 2z + 1/2$ avec $\varepsilon = 0.08476385681 \approx \beta(p)$

7.5 Calcul du ε-pseudoabscisse

Dans cette section, nous proposons trois algorithmes pour calculer l'ε-pseudoabscisse d'un polynôme donné. On rappelle que l'ε-pseudoabscisse est définie par

$$a_\varepsilon(p) = \max\{\mathrm{Re}(z) : z \in Z_\varepsilon(p)\}.$$

7.5.1 Tracer de pseudozéros

Une première méthode pour calculer le ε-pseudoabscisse est de tracer l'ensemble des ε-pseudozéros en utilisant l'algorithme 5.1. Une fois tracé cet ensemble, il suffit de tracer la droite verticale qui passe par le point le plus à droite de l'ensemble des ε-pseudozéros. Le ε-pseudoabscisse est l'abscisse d'un point d'intersection entre la droite tracée précédemment et l'axe réel.

Prenons comme exemple $p(z) = z^3+4z^2+6z+4$ avec $\varepsilon = 0.1$. Nous traçons l'ensemble des ε-pseudozéros (voir la figure 7.6) et nous trouvons que $a_\varepsilon(p) \approx -0.9$.

7.5.2 Un algorithme de dichotomie

Nous proposons maintenant un algorithme de dichotomie pour calculer $a_\varepsilon(p)$ avec une précision arbitraire. La clé de cet algorithme est le théorème 7.2 ci-dessous. Avant d'énoncer ce théorème, nous avons besoin du lemme suivant.

Lemme 7.1. *Pour tout point z_1 dans $Z_\varepsilon(p)$, il existe un point z_2 vérifiant* $\mathrm{Re}(z_1) = \mathrm{Re}(z_2)$ *et* $h_{p,\varepsilon}(z_2) = 0$.

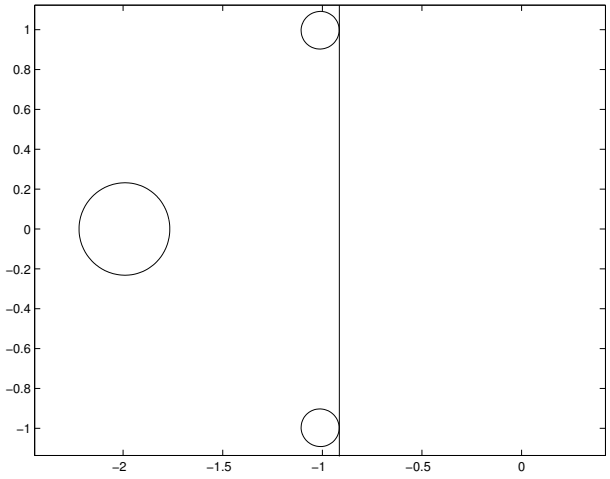


FIG. 7.6 – Ensemble des ε-pseudozéros de $p(z) = z^3 + 4z^2 + 6z + 4$ avec $\varepsilon = 0.1$

Démonstration. Nous pouvons prendre z_2 sur la frontière de l'intersection entre la ligne verticale passant par z_1 et l'ensemble $Z_\varepsilon(p)$ (c'est un ensemble compact). ■

Le résultat le plus important de cette sous-section est le théorème suivant. Il fournit une borne inférieure pour $a_\varepsilon(p)$.

Théorème 7.2. *Pour tout réel $x \geqslant a(p)$, les assertions suivantes sont équivalentes :*

(i) $x \leqslant a_\varepsilon(p)$;

(ii) *l'équation polynomiale*

$$h_{p,\varepsilon}(x, y) = 0, \tag{7.7}$$

admet une solution réelle y.

Démonstration. Commençons par montrer (i) $\Rightarrow$ (ii). Si $x = a_\varepsilon(p)$ alors on peut choisir un point z vérifiant $\mathrm{Re}(z) = a_\varepsilon(p)$. D'après le lemme 7.1, il est clair que $z = x + iy$ pour un y réel avec $h_{p,\varepsilon}(x, y) = 0$. Si $x < a_\varepsilon(p)$, alors par définition de $a_\varepsilon(p)$, il existe un point z_1 tel que $\mathrm{Re}(z_1) > x$ et $g(z_1) < \varepsilon$. La composante connexe de $Z_\varepsilon(p)$ contenant z_1 contient une racine z_2 de p d'après le théorème 5.3. Par conséquent, il existe un chemin continu dans cette composante connexe reliant z_1 et z_2 (voir la preuve du théorème 5.3). Mais comme $\mathrm{Re}(z_1) \geqslant x > \mathrm{Re}(z_2)$, ce chemin contient un point z_3 tel que $\mathrm{Re}(z_3) = x$. D'après le lemme 7.1, nous avons une solution pour l'équation (7.7).
Prouvons maintenant (ii) $\Rightarrow$ (i). Soit $y \in \mathbf{R}$ tel que $h_{p,\varepsilon}(x, y) = 0$. Cela implique que $x + iy \in Z_\varepsilon(p)$. D'après la définition de $a_\varepsilon(p)$, on obtient $x \leqslant a_\varepsilon(p)$. ■

Nous présentons maintenant un algorithme de dichotomie pour calculer une approximation de $a_\varepsilon(p)$. D'après la définition de $a_\varepsilon(p)$ et la proposition 7.2, nous savons que $a_\varepsilon(p)$ se situe dans l'intervalle $[a(p), \|p\| + \varepsilon]$. Notons x le milieu de cet intervalle. Nous pouvons calculer les solutions de l'équation $h_{p,\varepsilon}(x, y) = 0$, $y \in \mathbf{R}$ (ce calcul n'est pas nécessaire dans la pratique). Si une des solutions est réelle alors nous savons que $x \leqslant a_\varepsilon(p)$ d'après le

théorème 7.2, et nous remplaçons l'intervalle courant par le nouvel intervalle $[x, \|p\| + \varepsilon]$. Sinon, si $x > a_\varepsilon(p)$, nous remplaçons l'intervalle courant par $[a(p), x]$. Une implémentation de cette méthode est proposée par l'algorithme 7.2 où τ désigne la précision voulue pour la valeur de l'approximation de $a_\varepsilon(p)$.

Algorithme 7.2 Calcul de l'ε-pseudoabscisse par dichotomie

Entrée : un polynôme p, le paramètre ε et une tolérance τ
Sortie : un réel α vérifiant $|\alpha - a_\varepsilon(p)| \leqslant \tau$

$\gamma := a(p), \quad \delta := \|p\| + \varepsilon$
tant que $|\gamma - \delta| > \tau$ **faire**
  $x := \frac{\gamma+\delta}{2}$
  si l'équation $h_{p,\varepsilon}(x, y) = 0,\ y \in \mathbf{C}$ a une solution réelle **alors**
    $\gamma := x$
  sinon
    $\delta := x$
  fin si
fin tant que
retourne $\alpha = \frac{\gamma+\delta}{2}$

L'étape difficile est toujours de tester si le polynôme $H_x(y) = h_{p,\varepsilon}(x, y) = h_{2n}y^{2n} + \cdots + h_1 y + h_0$ a des racines réelles. Nous remarquons facilement que H_x est à coefficients réels par définition de $h_{p,\varepsilon}$. Comme précédemment, nous allons utiliser des suites de Sturm. Il est bien connu que le maximum des modules r des racines de H_x vérifie (voir [138])

$$r \leqslant \frac{\|H_x\|}{|h_{2n}|}.$$

Par conséquent, les possibles racines réelles de H_x appartiennent nécessairement à l'intervalle $[-r, r]$. Comme dans la section 7.3, on peut appliquer l'algorithme des suites de Sturm à H_x pour tester si l'équation $h_{p,\varepsilon}(x, y) = 0,\ y \in \mathbf{C}$ a une solution réelle.

Nous avons implémenté l'algorithme 7.2 en MAPLE. Ce choix est dû au fait que nous avons besoin de faire des manipulations formelles sur les polynômes.

Prenons le même exemple que dans la sous-section 7.5.1, c'est-à-dire, $p(z) = z^3 + 4z^2 + 6z + 4$ avec $\varepsilon = 0.1$ et $\tau = 0.00001$. On trouve que $a_\varepsilon(p) \approx -0.919901$ ce qui est plus précis que le résultat graphique donné précédemment.

Regardons un autre exemple en prenant $q(z) = z^5 + 5z^4 + 10z^3 + 10z^2 + 5z + 1$ et $\varepsilon = 0.001$. L'algorithme 7.2 donne $a_\varepsilon(q) = -0.719669$. La figure 7.7 représente le tracé de l'ensemble des ε-pseudozéros de q pour $\varepsilon = 0.001$, $\tau = 0.00001$ et donne une représentation graphique de $a_\varepsilon(p)$.

7.5.3 Un algorithme de type criss-cross

Nous présentons dans cette sous-section un algorithme de type criss-cross inspiré de [21, 29]. L'idée n'est plus comme dans la dichotomie de prendre le milieu de deux points.

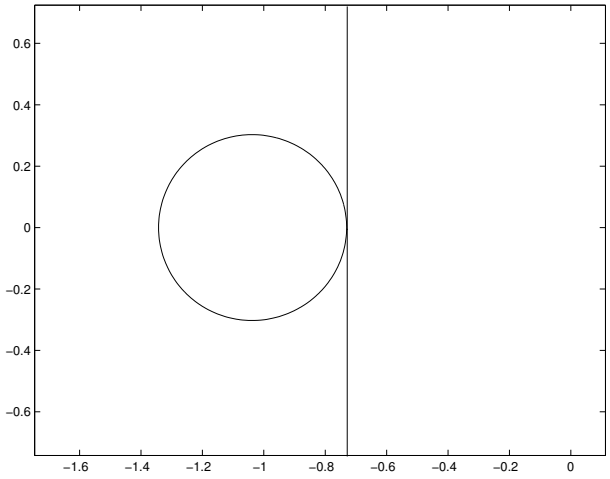


FIG. 7.7 – Ensemble des ε-pseudozéros de $q(z) = z^5 + 5z^4 + 10z^3 + 10z^2 + 5z + 1$ pour $\varepsilon = 0.001$

Il s'agit un fois un point donné dans le pseudozero de faire une recherche horizontale pour trouver le point le plus à droite sur la ligne dans ce pseudozéro. Une fois ce point trouver, on fait une recherche verticale et ainsi de suite.

Nous avons besoin tout d'abord du lemme suivant.

Lemme 7.2. *Pour tout* $x \in]a(p), a_\varepsilon(p)[$, *il existe* $y \in \mathbf{R}$ *tel que* $h_{p,\varepsilon}(x, y) < 0$.

Démonstration. Soit z_0 une racine de p telle que $\mathrm{Re}(z_0) = a(p)$ et $z_1 \in Z_\varepsilon(p)$ tel que $\mathrm{Re}(z_1) = a_\varepsilon(p)$. Soit $q \in \mathcal{M}_n$ vérifiant $q(z_1) = 0$ et$\|p - q\| \leqslant \varepsilon$. Définissons la famille de polynômes $p_t(z) = (1-t)p(z) + tq(z)$, $t \in [0, 1]$. Il est clair que $p_t \in \mathcal{M}_n$, $p_0(z) = p(z)$ et $p_1(z) = q(z)$. Introduisons la fonction $\varphi : [0, 1] \to \mathbf{R}$ définie par $t \mapsto a(p_t)$. La continuité de a sur $\mathcal{M}_n$ (voir la proposition 7.5) nous assure que la fonction φ est continue. Comme $\varphi(0) = a(p)$ et $\varphi(1) \geqslant a_\varepsilon(p)$, il existe $\bar{t} \in]0, 1[$ tel que $a(p_{\bar{t}}) = x$. De plus comme $\|p - p_{\bar{t}}\| < \varepsilon$, au moins une racine de $p_{\bar{t}}$ avec pour partie réelle x appartient à $Z'_\varepsilon(p)$, et ainsi $h(x, y) < 0$. ■

Algorithme 7.3 Calcul de l'ε-pseudoabscisse par un algorithme criss-cross

Entrée : un polynôme p, le paramètre ε

1: *Initialisation :* $x^1 = a(p)$ et $r = 1$
2: *Recherche verticale :* Trouver les intervalles ouverts $I_1^r, \ldots, I_{l_r}^r$ avec $h(x^r, y) < 0$ pour $y \in \cup_{k=1}^{l_r} I_k^r$
3: *Recherche horizontale :* pour chaque I_k^r, définir $\omega_k^r =$ milieu(I_k^r) et trouver la plus grande racine réelle x_k^r de la fonction $h(\cdot, \omega_k^r)$ pour $k = 1 : l_r$
4: Définir $x^{r+1} = \max\{x_k^r, k = 1, \ldots, l_r\}$, incrémenter r de un et retourner à l'étape 2.

Le théorème suivant prouve la convergence de l'algorithme 7.3.

Théorème 7.3. *L'algorithme criss-cross converge vers l'ε-pseudoabscisse* $a_\varepsilon(p)$.

Éléments de démonstration. La démonstration suit très exactement celle du théorème 3.2 de [29]. C'est pourquoi nous proposons juste des éléments de preuve. Nous notons $I_i^r = (l_i^r, u_i^r)$. Le fait que le nouvel itéré x^{r+1} soit un zéro de $h(\cdot, \omega_k^r)$ indique que $x^{r+1} \leqslant a_\varepsilon(p)$. Puisque $x^1 = a(p) \leqslant a_\varepsilon(p)$, on prouve par récurrence que $x^r \leqslant a_\varepsilon(p)$ pour tout r. Si à l'itération r, $x^r = a_\varepsilon(p)$, il n'y a rien à faire. Autrement, il existe un j tel que $l_j^r < u_j^r$ et donc $h_{p,\varepsilon}(x^r, \omega_j^r) < 0$. Cela implique que $x^{r+1} > x^r$. Par conséquent on déduit que la suite (x^r) est croissante et strictement bornée supérieurement par $a_\varepsilon(p)$ et bornée inférieurement par $a(p)$. Il résulte de tout cela que la suite (x^r) converge vers un réel x^∞ inférieur ou égale à $a_\varepsilon(p)$. Supposons que $x^\infty < a_\varepsilon(p)$. Dans ce cas, d'après le lemme 7.2, on déduit qu'il existe un intervalle ouvert dans lequel on a $h_{p,\varepsilon}(x^\infty, y) < 0$. On peut donc trouver un ω_k^r pour r et k suffisamment grand tel que $h_{p,\varepsilon}(x^\infty, \omega_k^r) \leqslant 0$. Cela implique que $x^{r+1} \geqslant x^\infty$. Cela contredit le fait que la suite (x^r) est strictement croissante. Ainsi nous avons $x^\infty = a_\varepsilon(p)$. ■

7.6 Conclusion

Dans ce chapitre, nous avons montré que les pseudozéros permettent de répondre efficacement à des problèmes de robustesse en théorie du contrôle et en automatique.

Les algorithmes existants pour calculer le rayon de stabilité sont des algorithmes matriciels prenant en entrée la matrice compagnon. Notre contribution a été de proposer un algorithme purement polynomial pour calculer ce rayon de stabilité.

Il n'y avait pas, à notre connaissance, d'algorithmes pour calculer le pseudoabscisse d'un polynôme. Notre contribution a été de modifier l'algorithme de calcul du rayon de stabilité pour obtenir un algorithme de calcul du pseudoabscisse.

7.7 Code MATLAB de l'algorithme 7.1

```
restart:
readlib(sturm):

stability_radius := proc(p,tau)

# p : the input polynomial
# tau : the wanted precision on the stability radius

local
  gama,      # lower bound for the stability radius
  delta,     # upper bound for the stability radius
  epsilon,   # incertainty on the polynomial coefficients
  h,         # the level function of the pseudozero set
  deg,       # the degree of the polynomial p
```

```
  R,          # the real of h are in the interval [-R,R]
  nb:         # number of real roots of the polynomial h

deg := degree(p,z):
gama := 0 :
delta := norm(p-z^deg,2,z):

while (evalf(abs(gama-delta)) > tau) do

   epsilon := evalf((delta + gama)/2):
   h := evalc(abs(subs(z=I*y,p)))^2 - epsilon^2*sum(y^(2*j),j=0..deg-1):
   h := simplify(expand(h,y)):
   R := evalf(norm(h,2,y)/abs(lcoeff(h,y))):
   nb := sturm(h,y,-R,R):

   if (nb > 0) then
      delta := epsilon:
   else
      gama := epsilon:
   fi:

od:

RETURN(evalf((delta + gama)/2)):
end:
```

7.8 Code MATLAB de l'algorithme 5.1

Il s'agit du code utilisé pour obtenir les figures précédentes.

```
function [] = pseudozero(P, epsilon)

[m,n] = size(P);

x= -3.5 : 1e-2 : 0.5;
y= -1.5  : 1e-2 : 1.5;

[X,Y] = meshgrid(x,y);
[r,s] = size(X);
Z = X + i.*Y;
modZ = abs(Z);

Pval = abs(polyval(P,Z));
un = ones(1,n-1);
```

```
Qval = (polyval(un,modZ.^2)).^(1/2);

for i=1:r
   for j=1:s
      Res(i,j) = Pval(i,j)/Qval(i,j);
   end
end

contour(x,y,Res,[epsilon epsilon],'r');
```

Chapitre 8

Zéros de polynômes d'intervalles

Dans ce chapitre, nous nous intéressons à la notion de polynômes d'intervalle. Il s'agit de polynômes dont les coefficients ne sont plus des nombres réels ou complexes mais des intervalles réels. Cela va nous permettre de modéliser des incertitudes sur les coefficients d'un polynôme. Dans la section 8.1, nous nous intéressons aux pseudozéros réels de polynômes réels avec des perturbations mesurées en norme infinie (voir la section 5.6 du chapitre 5). Dans la section 8.2, nous présentons les bases de l'arithmétique d'intervalle. Enfin, dans la section 8.3, nous présentons une formule calculable et un outil pour tracer les pseudozéros d'un polynôme d'intervalle.

8.1 Pseudozéros réels de polynômes en norme infinie

Dans cette section, nous introduisons l'ensemble des pseudozéros réels d'un polynôme pour des perturbations pondérées en norme ∞.

Nous utilisons les mêmes notations que précédemment. Pour $n \geqslant 1$, nous notons $\mathcal{P}_n(\mathbf{R})$ l'espace des polynômes à coefficients réels de degré au plus n. Soit p un polynôme de $\mathcal{P}_n(\mathbf{R})$ donc les coefficients sont notés p_i,

$$p(z) = \sum_{i=0}^{n} p_i z^i.$$

Nous identifierons le polynôme p avec le vecteur $(p_0, p_1, \ldots, p_n)^T$ de ses coefficients. Étant donné un vecteur $d := (d_0, \ldots, d_n)^T \in \mathbf{C}^{n+1}$, nous utiliserons la norme pondérée notée $\|\cdot\|_{\infty,d}$ définie par

$$\|p\|_{\infty,d} = \max_{i=0:n} \{|p_i|/|d_i|\}.$$

Le vecteur d permet de pondérer la perturbation selon des coefficients du polynôme.

Soit ε une borne sur l'incertitude des coefficients du polynôme. Un ε-voisinage de p est l'ensemble de tous les polynômes de $\mathcal{P}_n(\mathbf{R})$, suffisamment proches de p, c'est-à-dire,

$$N_\varepsilon^R(p) = \{\widehat{p} \in \mathcal{P}_n(\mathbf{R}) : \|p - \widehat{p}\|_{\infty,d} \leqslant \varepsilon\}. \tag{8.1}$$

L'ensemble des ε-pseudozéros p est alors l'ensemble de tous les zéros de tous les polynômes du ε-voisinage de p. Formellement, on a

$$Z_\varepsilon^R(p) = \left\{ z \in \mathbf{C} : \widehat{p}(z) = 0 \text{ for } \widehat{p} \in N_\varepsilon^R(p) \right\}. \tag{8.2}$$

Pour $\varepsilon = 0$, l'ensemble des pseudozéros $Z_0^R(p)$ n'est autre que l'ensemble des zéros de p que l'on note $Z(p)$.

On remarque facilement que l'ensemble des ε-pseudozéros réels $Z_\varepsilon^R(p)$ est symétrique par rapport à l'axe réel.

Proposition 8.1. *L'ensemble des ε-pseudozéros réels $Z_\varepsilon^R(p)$ est symétrique par rapport à l'axe réel.*

Démonstration. Soit $z \in Z_\varepsilon^R(p)$. Alors il existe $q \in N_\varepsilon^R(p)$ tel que $q(z) = 0$. Comme les coefficients de q sont réels, on en déduit que $q(\overline{z}) = 0$, si bien que $\overline{z} \in Z_\varepsilon^R(p)$. ■

Le théorème 8.1 suivant fournit une version calculable de l'ensemble des pseudozéros. Il est basé sur la preuve de [93, Theorem 5.1]. Il a été récemment prouvé par Karow [114] en utilisant des perturbations structurées de la matrice compagnon. Nous prouvons ici le résultat en restant dans le cadre polynomial. Étant donnés $x, y \in \mathbf{R}^{n+1}$, nous définissons par

$$d(x, \mathbf{R}y) = \inf_{\alpha \in \mathbf{R}} \|x - \alpha y\|_{1,d},$$

la distance du point $x \in \mathbf{R}^{n+1}$ à la droite $\mathbf{R}y = \{\alpha y, \alpha \in \mathbf{R}\}$. La norme $\|\cdot\|_{1,d}$ est définie pour $x \in \mathbf{R}^{n+1}$ par

$$\|x\|_{1,d} := \sum_{i=0}^{n} |d_i||x_i|.$$

Théorème 8.1. *L'ensemble des ε-pseudozéros réels de p vérifie*

$$Z_\varepsilon^R(p) = Z(p) \cup \left\{ z \in \mathbf{C} \backslash Z(p) : h(z) := d(G_R(z), \mathbf{R}G_I(z)) \geqslant \frac{1}{\varepsilon} \right\}, \tag{8.3}$$

où $G_R(z)$ et $G_I(z)$ sont respectivement les parties réelles et imaginaires de

$$G(z) = \frac{1}{p(z)}(1, z, \ldots, z^n)^T, \ z \in \mathbf{C} \backslash Z(p).$$

Démonstration. Soit $z \in Z_\varepsilon^R(p)$. Si $p(z) = 0$ alors $z \in Z(p)$ sinon il existe $q \in N_\varepsilon^R(p)$ tel que $q(z) = 0$. Dans ce cas, nous avons $p(z) = p(z) - q(z) = (p - q)^T \underline{z}$, avec $\underline{z} = (1, z, \ldots, z^n)^T$. On en déduit que $1 = (p-q)^T G(z)$. Ainsi, nous avons $1 = (p-q)^T G_R(u) + i(p-q)^T G_I(u)$ et donc

$$\begin{cases} (p-q)^T G_R(u) = 1, \\ (p-q)^T G_I(u) = 0. \end{cases}$$

Par conséquent, $\|p - q\|_{\infty,d} \|G_R(u) - \alpha G_I(u)\|_{1,d} \geqslant 1$, pour tout $\alpha \in \mathbf{R}$. On conclut donc que

$$d(G_R(u), \mathbf{R}G_I(u)) \geqslant \frac{1}{\|p - q\|_{\infty,d}} \geqslant \frac{1}{\varepsilon}.$$

Réciproquement, soit $z \in Z(p) \cup \left\{ z \in \mathbf{C} \backslash Z(p) : d(G_R(z), \mathbf{R}G_I(z)) \geqslant \frac{1}{\varepsilon} \right\}$. Si z appartient à $Z(p)$ alors il appartient aussi à $Z_\varepsilon^R(p)$. Autrement z vérifie $d(G_R(z), \mathbf{R}G_I(z)) \geqslant 1/\varepsilon$. D'après un théorème de dualité (voir [132, p.119]), il existe un vecteur $u \in \mathbf{R}^{n+1}$ avec $\|u\|_{\infty,d} = 1$ satisfaisant

$$u^T G_R(z) = d(G_R(z), \mathbf{R}G_I(z)) \quad \text{and} \quad u^T G_I(z) = 0.$$

Considérons alors le polynôme réel

$$q = p - \frac{u}{d(G_R(z), \mathbf{R}G_I(z))}.$$

Il vérifie

$$q(z) = p(z) - \frac{u^T \underline{z}}{d(G_R(z), \mathbf{R}G_I(z))} = p(z) - \frac{p(z) u^T G(z)}{d(G_R(z), \mathbf{R}G_I(z))} = 0.$$

De plus on a $\|q - p\|_{\infty,d} = 1/d(G_R(z), \mathbf{R}G_I(z))$. Il s'en suit $\|p - q\|_{\infty,d} \leqslant \varepsilon$. ■

Pour calculer l'ensemble des ε-pseudozéros réels $Z_\varepsilon^R(p)$, il nous suffit de savoir évaluer la distance $d(G_R(z), \mathbf{R}G_I(z))$. On peut montrer (voir [114, Prop. 7.7.2]) que

$$d(x, \mathbf{R}y) = \begin{cases} \min\limits_{\substack{i=0:n \\ y_i \neq 0}} \|x - (x_i/y_i)y\|_{1,d} & \text{si } y \neq 0, \\ \|x\|_{1,d} & \text{si } y = 0. \end{cases}$$

L'ensemble des pseudozéros réels est donc l'intérieur de l'ensemble défini par la ligne de niveau de la fonction h définie par la relation 8.3. On peut alors tracer l'ensemble des ε-pseudozéros réels grâce à l'algorithme 8.1 suivant.

Algorithme 8.1 Calcul de l'ensemble des ε-pseudozéros réels (MATLAB)

Entrée : le polynôme p et une incertitude ε
Sortie : l'ensemble des ε-pseudozéros réels dans le plan complexe

1: On grille un carré contenant les ε-pseudozéros de p avec la commande MATLAB `meshgrid`.
2: On calcule $h(z)$ pour tous les points z de la grille.
3: On trace les lignes de niveau $h(z) = 1/\varepsilon$ avec la commande MATLAB `contourf`.

La proposition suivante montre que le nombre de composantes connexes de chaque ensemble de pseudozéros contient au moins une racine du polynôme (voir le théorème 5.3 du chapitre 5 pour la preuve dans le cas complexe mais qui se transpose directement pour le cas réel).

Proposition 8.2 (Mosier [143]). *Étant donné $p \in \mathcal{P}_n(\mathbf{R})$ de degré n, supposons que l'ensemble des pseudozéros réels $Z_\varepsilon^R(p)$ est borné. Si $q \in N_\varepsilon^R(p)$, alors p et q ont le même nombre de racines, en comptant les multiplicités, dans chaque composante connexe de $Z_\varepsilon^R(p)$. De plus, il y a au moins une racine de p dans chaque composante connexe de $Z_\varepsilon^R(p)$.*

Stetter [185] a prouvé un résultat plus précis (voir le théorème 5.5 du chapitre 5 pour la preuve).

Proposition 8.3 (Stetter [185, Thm. 3.3]). *Soit $p \in \mathcal{P}_n(\mathbf{R})$ un polynôme réel de degré n. Soit Z_μ une composante connexe de $Z_\varepsilon^R(p)$ tel que p ait une seule racine dans Z_μ. Alors Z_μ vérifie soit $Z_\mu \subset \mathbf{R}$ ou $Z_\mu \cap \mathbf{R} = \emptyset$.*

Comme nous venons de le voir, l'ensemble des pseudozéros réels est intimement lié à la fonction d. Cette fonction peut être discontinue. C'est l'objet du lemme suivant.

Lemme 8.1 (Hinrichsen et Kelb [89]). *La fonction*

$$d : \mathbf{R}^{n+1} \times \mathbf{R}^{n+1} \to \mathbf{R}_+, \quad (x, y) \mapsto d(x, \mathbf{R}y)$$

est continue en tous les points (x, y) tel que $y \neq 0$ ou $x = 0$, et discontinue en tous les points $(x, 0) \in \mathbf{R}^{n+1} \times \mathbf{R}^{n+1}$, $x \neq 0$.

Ce lemme établit que des discontinuités surviennent lorsque le vecteur y s'annule. Dans notre cas, les discontinuités surviennent quand $G_I(z) = 0$ où $G_I(z)$ est la partie imaginaire de

$$G(z) = \frac{1}{p(z)}(1, z, \ldots, z^n)^T.$$

On en déduit alors que G_I s'annule pour $z \in \mathbf{R}$, c'est-à-dire sur l'axe réel. Ceci explique pourquoi les commandes MATLAB `contourf` et `meshc` peuvent donner de mauvais résultats le long de l'axe réel. Bien entendu, si aucun des zéros de notre polynôme n'est réel, alors on obtient de très bons résultats pour le tracé.

Afin d'éviter ce problème de discontinuité, on peut calculer spécifiquement les pseudozéros sur l'axe réel. Pour ce faire, on utilise le lemme suivant.

Lemme 8.2. *Soit $z \in \mathbf{R}$, z appartient à $Z_\varepsilon^R(p)$ si et seulement si z appartient à $Z_\varepsilon(p)$ (ensemble des pseudozéros complexes).*

Démonstration. Ceci est vrai car la formule donnant le polynôme le plus proche ayant une racine donnée dans la preuve du théorème 5.2 (page 73) fournit bien un polynôme à coefficients réels si z est réel. ∎

8.2 Introduction à l'arithmétique d'intervalles

Cette partie est très largement inspirée des articles suivants [78, 167]. Le principe de l'analyse par intervalles est de calculer avec des intervalles de nombres réels plutôt qu'avec les nombres réels eux-mêmes. Tandis que l'arithmétique flottante est affectée par les erreurs d'arrondis et donc peut conduire à des résultats imprécis et faux, l'analyse par intervalles a l'avantage de donner des bornes rigoureuses pour la solution exacte. Une telle analyse se révèle très utile quand par exemple les paramètres ne sont connus qu'avec une certaine incertitude. Dans ce cas, on peut implémenter des algorithmes en utilisant

l'arithmétique d'intervalles pour les paramètres incertains afin de produire un intervalle qui contient tous les résultats possibles.

Si la borne supérieure et la borne inférieure dans l'intervalle peuvent être arrondies vers le bas et vers le haut respectivement alors les calculs en précision finie peuvent être effectués en utilisant des intervalles incluant chaque nombre réel. Si l'arithmétique utilisée est l'arithmétique flottante satisfaisant la norme IEEE 754, cela est alors possible. En effet, cette norme impose que le résultat calculé soit l'arrondi du résultat exact avec quatre modes d'arrondis possible : vers $+\infty$, vers $-\infty$, au plus près et vers 0. L'utilisation de l'arrondi vers $+\infty$ pour la borne supérieure et de l'arrondi vers $-\infty$ pour la borne inférieure permet d'implanter une arithmétique d'intervalle.

L'idée de borner les erreurs d'arrondis en utilisant des intervalles est apparue dans les années 50. Néanmoins, cette idée n'a connu un essor qu'à partir de la publication d'un livre sur le sujet par Moore [141] en 1966. Depuis le thème a connu un immense développement [107, 102, 3, 141, 142, 144, 145, 170]. On peut citer par exemple le fait que plus de la moitié des problèmes posés par Trefethen dans son « 100-Digits Challenge » ont été résolus via l'arithmétique d'intervalle (voir [16]).

Dans ce qui suit, un intervalle sera représenté par ses extrémités

$$\boldsymbol{x} = [\underline{x}; \overline{x}] = \{x \in \mathbf{R} : \underline{x} \leqslant x \leqslant \overline{x}\}.$$

Tout nombre réel sera confondu avec l'intervalle $[x; x]$ correspondant. Les quantités intervalles seront notées en caractères gras. L'ensemble des intervalles de $\mathbf{R}$ sera noté $\mathbf{IR}$. On notera par $\mathrm{mid}(\boldsymbol{x})$ le milieu de l'intervalle $\boldsymbol{x} = [\underline{x}; \overline{x}]$,

$$\mathrm{mid}(\boldsymbol{x}) = (\overline{x} + \underline{x})/2,$$

et par $w(\boldsymbol{x})$ la largeur de $\boldsymbol{x}$,

$$w(\boldsymbol{x}) = \overline{x} - \underline{x}.$$

Le rayon de $\boldsymbol{x}$ noté $\mathrm{rad}(\boldsymbol{x})$ est la moitié de la largeur de $\boldsymbol{x}$. On définie la mignitude d'un intervalle $\boldsymbol{x}$ par

$$\mathrm{mig}(\boldsymbol{x}) = \min\{|x| : x \in \boldsymbol{x}\},$$

et la magnitude par

$$|\boldsymbol{x}| = \mathrm{mag}(\boldsymbol{x}) = \max\{|x| : x \in \boldsymbol{x}\}.$$

Ces deux quantités peuvent être calculées en utilisant les extrémités de l'intervalle,

$$\begin{aligned} \mathrm{mag}(\boldsymbol{x}) &= \max\{|\underline{x}|, |\overline{x}|\}, \\ \mathrm{mig}(\boldsymbol{x}) &= \begin{cases} \min\{|\underline{x}|, |\overline{x}|\} & \text{si } 0 \notin \boldsymbol{x}, \\ 0 & \text{sinon.} \end{cases} \end{aligned}$$

Il est possible d'étendre les opérations usuelles aux intervalles par la formule suivante : étant donnés deux intervalles $\boldsymbol{x}$, $\boldsymbol{y}$ et $\diamond \in \{+, -, \times, /\}$, on définit

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

On peut implémenter ces opérations de la manière suivante :

$$\begin{aligned}
\boldsymbol{x}+\boldsymbol{y} &= [\underline{x}+\underline{y}; \overline{x}+\overline{y}],\\
\boldsymbol{x}-\boldsymbol{y} &= [\underline{x}-\overline{y}; \overline{x}-\underline{y}],\\
\boldsymbol{x}\times\boldsymbol{y} &= [\min\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}; \max\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}],\\
1/\boldsymbol{x} &= [1/\overline{x}; 1/\underline{x}],\\
\boldsymbol{x}/\boldsymbol{y} &= \boldsymbol{x}\times 1/\boldsymbol{y}.
\end{aligned}$$

On peut ensuite définir un vecteur d'intervalles comme un vecteur dont les composantes sont des intervalles réels et donc définir l'espace de dimension n des vecteurs d'intervalles que l'on note $\mathbf{IR}^n$. De la même façon, on peut aussi définir une matrice d'intervalles comme une matrice dont les composantes sont des intervalles. L'ensemble des matrices d'intervalles est noté $\mathbf{IR}^{m\times n}$.

Plusieurs outils ont été développés utilisant l'arithmétique d'intervalles. Selon [76], on peut les diviser en deux partie : d'un coté il y a les librairies pour les logiciels de calcul scientifique ou de calcul formel et de l'autre les librairies pour les langages de programmation. Les packages pour les logiciels scientifiques sont :

- INTLAB pour MATLAB (voir [171]) ;
- Mathematica ;
- intpakX pour MAPLE.

Les librairies les plus connus intégrant l'arithmétique d'intervalle pour les langages de programmation sont

- XSC (eXtended Scientific Computing) ;
- l'arithmétique « range » ;
- MPFI (Multiple Precision Floating-point Interval arithmetics library).

À notre connaissance, aucun de ces packages ou librairies n'est capable de calculer les zéros d'un polynôme d'intervalle. Dans la suite de chapitre, nous proposons une méthode et un outil pour calculer ces zéros.

8.3 Polynômes d'intervalles

Un *polynômes d'intervalles* est un polynôme dont les coefficients sont des intervalles réels. Nous noterons $\mathbf{IR}[z]$ l'ensemble des polynômes d'intervalles et $\mathbf{IR}_n[z]$ l'ensemble des polynômes d'intervalles de degré au plus n. Soit $\mathbf{p}$ un élément de $\mathbf{IR}_n[z]$. On peut l'écrire sous la forme

$$\boldsymbol{p}(z) = \sum_{i=0}^{n} [a_i, b_i] z^i.$$

Les zéros d'un polynôme d'intervalles sont les éléments de l'ensemble noté $\mathbf{Z}(\boldsymbol{p})$ défini par

$$\mathbf{Z}(\boldsymbol{p}) := \{z \in \mathbf{C} : \text{il existe } m_i \in [a_i, b_i], i = 0 : n \text{ tel que } \sum_{i=0}^{n} m_i z^i = 0\}.$$

Nous supposerons à partir de maintenant que l'intervalle dominant $[a_n, b_n]$ ne contient pas 0. En effet, si tel n'est pas le cas, l'ensemble $\mathbf{Z}(\boldsymbol{p})$ devient non borné. Notre but est ici de calculer $\mathbf{Z}(\boldsymbol{p})$. Pour ce faire, nous avons besoin d'introduire le polynôme central p^c définie par

$$p^c(z) = \sum_{i=0}^{n} c_i z^i,$$

où $c_i = \mathrm{mid}([a_i, b_i])$. Posons aussi $d_i := (b_i - a_i)/2$.

Proposition 8.4. *Avec les notations précédentes, nous avons*

$$\mathbf{Z}(\boldsymbol{p}) = Z_\varepsilon^R(p^c) \text{ pour } \varepsilon = 1.$$

Démonstration. Soit $z \in \mathbf{C}$ appartenant à $\mathbf{Z}(\boldsymbol{p})$. Cela signifie qu'il existe $m_i \in [a_i, b_i]$ tel que $m(z) = \sum_{i=0}^n m_i z^i = 0$. Soit p^c le polynôme central défini comme précédemment. Nous avons alors

$$\|m_i - p_i^c\|_{\infty,d} = \max_{i=0:n}\{|m_i - p_i^c|/|d_i|\} \leqslant 1,$$

puisque $d_i := (b_i - a_i)/2$ et $p_i^c = (b_i + a_i)/2$. Ainsi le polynôme m qui s'annule en z est à une distance inférieure ou égale à 1 du polynôme central p_c (pour la norme pondérée considérée). Il s'en suit que z appartient à $Z_\varepsilon^R(p^c)$ avec $\varepsilon = 1$.

Réciproquement, soit z appartenant à $Z_\varepsilon^R(p^c)$ avec $\varepsilon = 1$. Cela signifie qu'il existe un polynôme q tel que $\|q - p^c\|_{\infty,d} \leqslant 1$. Alors on a $\max_{i=0:n}\{|q_i - p_i^c|/|d_i|\} \leqslant 1$. Un calcul simple montre alors que $a_i \leqslant q_i \leqslant b_i$ et donc que $z \in \mathbf{Z}(\boldsymbol{p})$. ■

Maintenant nous avons une formule calculable pour tracer l'ensemble des zéros d'un polynôme d'intervalle. Le seul problème restant est de trouver *a priori* une grille qui contienne l'ensemble des zéros du polynôme d'intervalle. C'est l'objet du lemme suivant.

Lemme 8.3. *Soit* $\boldsymbol{p}(z) = \sum_{i=0}^n [a_i, b_i] z^i$ *un polynôme d'intervalle et*

$$R := 1 + \frac{\max_{i=0:n}\{\max\{|a_i|, |b_i|\}\}}{\min\{|a_n|, |b_n|\}}.$$

Alors on a

$$\mathbf{Z}(\boldsymbol{p}) \subset B(O, R),$$

où $B(O, R)$ *est la boule du plan complexe* $\mathbf{C}$ *de centre* O *et de rayon* R.

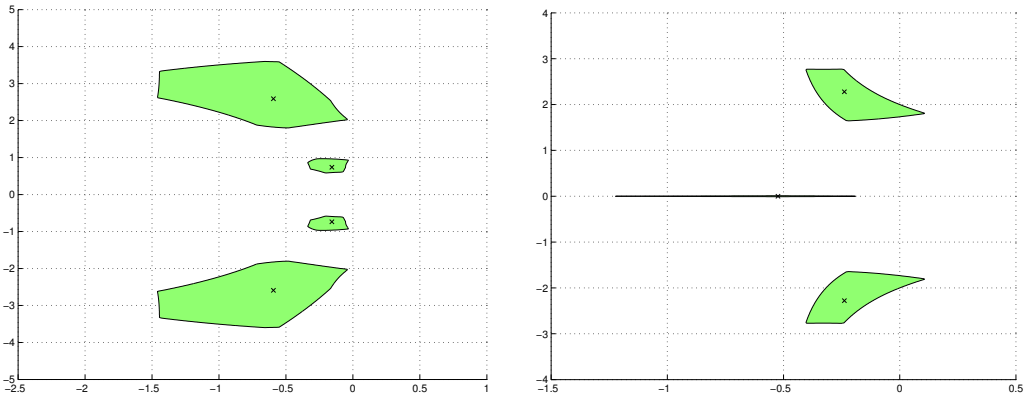
Démonstration. Soient $\{z_j\}_{j=1:n}$ les racines d'un polynôme p comptées avec leurs multiplicités et posons $r = \max_j |z_j|$. Il est bien connu (voir [138, p.154]) que

$$r \leqslant 1 + \frac{\max\{|p_{n-1}|, |p_{n-2}|, \ldots, |p_0|\}}{|p_n|}. \tag{8.4}$$

Soit z un élément de $\mathbf{Z}(\mathbf{p})$. Il existe alors un polynôme $m(z) = \sum_{i=0}^n m_i z^i$ vérifiant $m(z) = 0$ et $a_i \leqslant m_i \leqslant b_i$ pour $i = 0 : n$. Il est facile de voir que $|m_i| \leqslant \max\{|a_i|, |b_i|\}$ et $|p_n| \geqslant \min\{|a_n|, |b_n|\}$. En utilisant la relation 8.4, on obtient

$$|z| \leqslant 1 + \frac{\max_{i=0:n}\{\max\{|a_i|, |b_i|\}\}}{\min\{|a_n|, |b_n|\}}.$$

■



(a) $\boldsymbol{p}(z) = [1,2]z^4 + [3/2,3]z^3 + [10,14]z^2 + [3,5\sqrt{2}]z + [5,7]$

(b) $\boldsymbol{q}(z) = z^3 + z^2 + [3,8]z + [1.5,4]$

FIG. 8.1 – Ensemble des zéros des polynômes d'intervalle $\boldsymbol{p}(z)$ et $\boldsymbol{q}(z)$

8.3.1 Présentation de PSIP

PSIP (Pseudozero Set of Interval Polynomials) est une interface graphique qui intègre des routines MATLAB pour calculer et tracer les zéros d'un polynôme d'intervalles. Le code a été écrit et testé avec MATLAB versions 6.5 (R13).

Cette interface graphique intègre des facilités intéressantes pour tracer les zéros d'un polynôme d'intervalles. On peut par exemple zoomer sur des zones et changer interactivement le pas discrétisation afin de raffiner le tracé.

Un des inconvénients de cet outil est qu'il suppose que l'intervalle dominant du polynôme d'intervalles ne contient pas 0, puisque, sinon, l'ensemble des pseudozéros est non borné. De plus, nous avons essayé de traiter à part les discontinuité sur l'axe réel mais il reste toujours des problèmes lors de la visualisation proche de cet axe.

8.4 Conclusion

Dans ce chapitre, nous avons présenté une formule calculable pour l'ensemble des pseudozéros réels. Cette formule nous a permis de trouver une formule calculable pour l'ensemble des zéros d'un polynôme d'intervalle. Nous avons alors développé une interface graphique pour MATLAB afin de tracer cet ensemble.

Il serait intéressant de pouvoir tracer l'ensemble de manière certifiée. L'utilisation de l'algorithme SIVIA de Jaulin et Walter [107] semble être une bonne perspective.

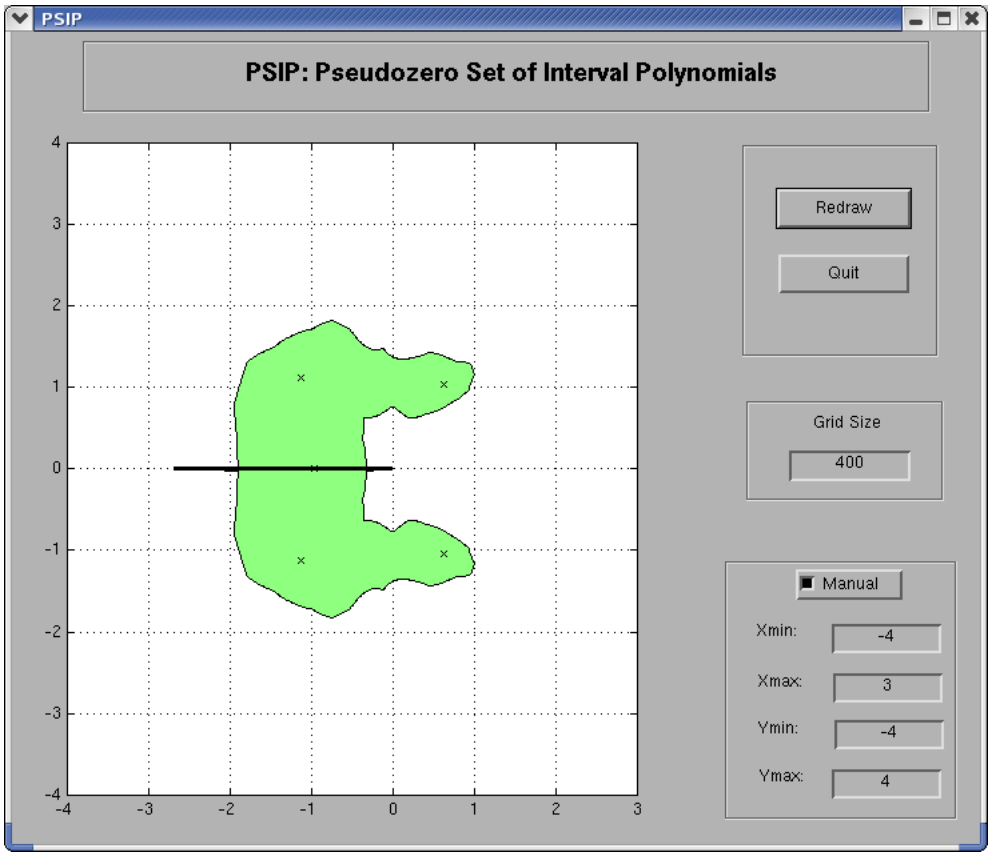


FIG. 8.2 – Interface graphique de PSIP avec les zéros du polynôme d'intervalle $\boldsymbol{p}(z) = z^5 + [1.20, 2.73]z^4 + [1.14, 3.15]z^3 + [0.20, 2.35]z^2 + [1.52, 6.21]z + [0.15, 7.11]$

Chapitre 9

Pseudozéros de polynômes à plusieurs indéterminées

Les polynômes à plusieurs variables apparaissent dans presque tous les champs du calcul scientifique et de l'ingénierie comme on peut le voir en lisant le « Computer Algebra Handbook » [68] et dans [53]. Les applications nécessitent de résoudre des équations polynomiales ou des systèmes polynomiaux contenant un très grand nombre d'indéterminées. Les relations entre les applications industrielles et la nécessité de résoudre des systèmes polynomiaux a été étudié par le projet européen FRISCO. Ils donnent une liste des champs majeurs où la résolution de systèmes polynomiaux est nécessaire : géométrie algorithmique et modélisation, mécanique, traitement du signal, ingénierie civile, robotique, simulation. La très large utilisation des systèmes polynomiaux nécessite d'avoir des algorithmes de résolution stables et efficaces. Actuellement il y a principalement deux grandes approches de résolution : l'une symbolique et l'autre numérique. L'approche symbolique est soit basée sur la théorie des bases de Gröbner soit sur la théorie des résultants. Pour l'approche numérique, on utilise plutôt des méthodes itératives telles que la méthode de Newton (souvent des versions améliorées) ou bien des méthodes d'homotopie. Récemment, des méthodes hybrides, combinant à la fois des méthodes symboliques et numériques sont apparues (voir le chapitre « Hybrid Methods » de Kaltofen dans [68, p.112-128]).

La majeure partie des articles cités précédemment considèrent seulement les pseudozéros de polynômes à une indéterminée. Les cas avec plusieurs indéterminées semblent avoir été très peu regardés excepté par Stetter [185, 186], par Hoffman, Madden et Zhang [98] et Corless, Kai et Watt [40]. De plus, le cas à plusieurs indéterminées a seulement été traité dans le cas de polynômes (et de systèmes polynomiaux) à coefficients complexes. Dans ce chapitre, nous considérons des systèmes ou les polynômes sont à coefficients réels et tels que tous les polynômes de tous les systèmes perturbés aient aussi des coefficients réels. Nous fournissons une formule explicite calculable pour tracer l'ensemble des pseudozéros et nous étudions différentes méthodes pour les visualiser.

Le chapitre est organisé de la façon suivante. Dans la section 9.1, nous rappelons les notations et des résultats de base en algèbre linéaire et en calcul formel. Dans la section 9.2, nous rappelons les résultats existants sur les pseudozéros de polynômes à coefficients complexes. Dans la section 9.3, nous étudions les pseudozéros réels et nous établissons un critère simple pour calculer cet ensemble. Enfin, dans la section 9.4, nous présentons différentes méthodes pour visualiser ces ensembles.

9.1 Notation

Nous rappelons les notations utilisées par Stetter [186]. Un *monôme* de n indéterminées $z_1, \ldots, z_n$ est le produit

$$z^j := z_1^{j_1} \cdots z_n^{j_n}, \quad \text{avec } j = (j_1, \ldots, j_n) \in \mathbf{N}^n;$$

j est l'*exposant* et $|j| := \sum_{\sigma=1}^{n} j_\sigma$ est le *degré* du monôme z^j.

Définition 9.1. Un *polynôme complexe (réel)* à n indéterminées est une combinaison linéaire finie de monômes à n indéterminées à coefficients dans $\mathbf{C}$ (dans $\mathbf{R}$),

$$p(z) = p(z_1, \ldots, z_n) = \sum_{(j_1,\ldots,j_n)\in J}^{n} a_{j_1\cdots j_n} z_1^{j_1} \cdots z_n^{j_n} = \sum_{j\in J} a_j z^j.$$

L'ensemble $J \subset \mathbf{N}^n$ qui contient les exposants de tous les monômes du polynôme p est appelé le *support* de p. Le *degré total* de p est l'entier $\deg(p) := \max_{j\in J} |j|$. On notera $\mathcal{P}^n(\mathbf{C})$ ($\mathcal{P}^n(\mathbf{R})$) l'ensemble des polynômes complexes (réels) à n indéterminées. Quand le domaine des coefficients est implicite ou n'est pas important nous utiliserons $\mathcal{P}^n$ à la place. La notation $\mathcal{P}_d^n \subset \mathcal{P}^n$ représente l'ensemble des polynômes à n indéterminées de degré total $\leqslant d$. Comme nous manipulerons très souvent les polynômes sous la forme d'un vecteur de leurs coefficients, nous emploierons largement les notations de l'algèbre linéaire. Nous noterons le polynôme par le vecteur de ses coefficients $a = (\ldots, a_j, \ldots, j \in J)^T$ et ses monômes par le vecteur $\mathbf{z} = (\ldots, z^j, \ldots, j \in J)^T$.

Soit $p = \sum_{j\in J} a_j z^j \in \mathcal{P}^n(\mathbf{K})$ avec $\mathbf{K} = \mathbf{R}$ ou $\mathbf{C}$ un polynôme à n indéterminées et J son support. Nous notons $|J|$ le nombre d'éléments de J. Si $|J| = M$ et $\|\cdot\|$ une norme sur $\mathbf{K}^M$, nous noterons $\|p\|$ la norme du vecteur $a = (\ldots, a_j, \ldots, j \in J)$, c'est-à-dire,

$$\|p\| := \|(\ldots, a_j, \ldots, j \in J)^T\|.$$

Étant donné $\varepsilon > 0$, le ε-voisinage $N_\varepsilon(p)$ du polynôme $p \in \mathcal{P}^n(\mathbf{K})$ est l'ensemble de tous les polynômes de $\mathcal{P}^n(\mathbf{K})$, suffisamment proche de p, c'est-à-dire, l'ensemble des polynômes $\widetilde{p} = \sum_{j\in\widetilde{J}} \widetilde{a}_j z^j \in \mathcal{P}^n(\mathbf{K})$ avec un support $\widetilde{J} \subset J$ et $\|\widetilde{p} - p\| \leqslant \varepsilon$.

Étant donnée une norme $\|\cdot\|$ sur $\mathbf{K}^N$ avec $\mathbf{K} = \mathbf{R}$ ou $\mathbf{C}$, on définit sa *norme duale*, notée $\|\cdot\|_*$, par

$$\|x\|_* := \sup_{y\neq 0} \frac{|y^T x|}{\|y\|} = \sup_{\|y\|=1} |y^T x|.$$

Normes	Normes duales
$\|x\|_1 := \sum_j \|x_j\|$	$\|x\|_1^* = \max_j \|x_j\| = \|x\|_\infty$
$\|x\|_2 := (\sum_j \|x_j\|^2)^{1/2}$	$\|x\|_2^* = (\sum_j \|x_j\|^2)^{1/2} = \|x\|_2$
$\|x\|_\infty := \max_j \|x_j\|$	$\|x\|_\infty^* = \sum_j \|x_j\| = \|x\|_1$

TAB. 9.1 – Norme duale des normes classiques sur $\mathbf{K}^N$

Le tableau 9.1 représente les normes classiques sur $\mathbf{K}^N$ et leurs normes duales respectives. Étant donné un vecteur $x \in \mathbf{K}^N$, il existe un vecteur $y \in \mathbf{K}^N$ vérifiant $\|y\| = 1$ et $x^T y = \|x\|_*$ (voir par exemple [87, p.107] ou [99, p. 278]). Ce vecteur y est appelé le *vecteur dual* de x.

Définition 9.2. Un élément $z \in \mathbf{K}^n$ est un *ε-pseudozéro* du polynôme $p \in \mathcal{P}^n$ si c'est le zéro d'un polynôme $\widetilde{p}$ de $N_\varepsilon(p)$.

Définition 9.3. L'*ensemble des ε-pseudozéros* du polynôme $p \in \mathcal{P}^n$ (noté $Z_\varepsilon(p)$) est l'ensemble de tous les ε-pseudozéros,

$$Z_\varepsilon(p) := \{z \in \mathbf{K}^n : \exists \widetilde{p} \in N_\varepsilon(p), \quad \widetilde{p}(z) = 0\}.$$

Cette définition soulève trois problèmes importants.

- Pour p avec des coefficients réels a_j, on doit spécifier si $N_\varepsilon(p)$ se restreint aux polynômes à coefficients réels ou non. Cependant, il semble naturel que l'on perturbe un polynôme à coefficients réels par des polynômes eux aussi à coefficients réels.
- On peut s'intéresser ou bien aux pseudozéros réels ou complexes.
- L'ensemble des pseudozéros $Z_\varepsilon(p)$ ne peut être calculer directement car ce sont les zéros d'un nombre infini de polynômes.

Il s'agit des questions auxquelles nous allons tacher de répondre dans ce chapitre. Nous pouvons étendre les définitions précédentes au cas des *systèmes polynomiaux*

$$P = \{p_1, \ldots, p_k\}, \quad k \in \mathbf{N}.$$

Nous considérons fréquemment ce système comme un vecteur de polynômes

$$P(z) = \begin{pmatrix} p_1(z) \\ \vdots \\ p_k(z) \end{pmatrix}.$$

Étant donnés $\varepsilon > 0$ et un système polynomial $P = \{p_1, \ldots, p_k\}$, $k \in \mathbf{N}$, le ε-voisinage $N_\varepsilon(P)$ est l'ensemble des systèmes polynomiaux $\widetilde{P} = \{\widetilde{p_1}, \ldots, \widetilde{p_k}\}$ suffisamment proche de P, c'est-à-dire $\widetilde{p_j} \in N_\varepsilon(p_j)$ for $j = 1, \ldots, k$.

Définition 9.4. Un élément $z \in \mathbf{K}^n$ est un *ε-pseudozéros* du système polynomial P si c'est un zéro d'un système polynomial $\widetilde{P}$ de $N_\varepsilon(P)$.

Définition 9.5. L'*ensemble des ε-pseudozéros* d'un système polynomial P (noté $Z_\varepsilon(P)$) est l'ensemble des tous les ε-pseudozéros,

$$Z_\varepsilon(P) := \{z \in \mathbf{K}^n : \exists \widetilde{P} \in N_\varepsilon(P), \quad \widetilde{P}(z) = 0\}.$$

9.2 Pseudozéros de polynômes à plusieurs indéterminées à coefficients complexes

Le théorème 9.1 ci-dessous fournit une formule explicite pour le calcul de l'ensemble des ε-pseudozéros.

Théorème 9.1 (Stetter [186]). *L'ensemble des ε-pseudozéros complexes de $p = \sum_{j\in J} a_j z^j \in \mathcal{P}^n(\mathbf{C})$ vérifie*

$$Z_\varepsilon(p) = \left\{ z \in \mathbf{C}^n : g(z) := \frac{|p(z)|}{\|\mathbf{z}\|_*} \leqslant \varepsilon \right\},$$

où $\mathbf{z} := (\ldots, |z|^j, \ldots, j \in J)^T$.

Démonstration. Si $z \in Z_\varepsilon(p)$ alors il existe $\widetilde{p} \in \mathcal{P}^n$ tel que $\widetilde{p}(z) = 0$ et $\|p - \widetilde{p}\| \leqslant \varepsilon$. D'après l'inégalité de Hölder généralisée $|x^T y| \leqslant \|x\|\|y\|_*$, on a

$$|p(z)| = |p(z) - \widetilde{p}(z)| = \Big| \sum_{j\in J} (p_j - \widetilde{p}_j) z^j \Big| \leqslant \|p - \widetilde{p}\| \|\mathbf{z}\|_*.$$

On en déduit donc que $|p(z)| \leqslant \varepsilon \|\mathbf{z}\|_*$.
Réciproquement, soit $u \in \mathbf{C}$ tel que $|p(u)| \leqslant \varepsilon \|\mathbf{u}\|$ où $\mathbf{u} := (\ldots, |u|^j, \ldots, j \in J)$. Le vecteur dual d de $\mathbf{u}$ satisfait $d^T \mathbf{u} = \|\mathbf{u}\|_*$ et $\|d\| = 1$. Introduisons les polynômes r et p_u définis par

$$\begin{aligned} r(z) &= \sum_{j\in J} d_j z^j \\ p_u(z) &= p(z) - \frac{p(u)}{r(u)} r(z). \end{aligned}$$

Le polynôme p_u est (au sens de la norme $\|\cdot\|$) le polynôme le plus proche de p ayant u comme racine. Il est clair que $r(u) = d^T \mathbf{u} = \|\mathbf{u}\|_*$. Par conséquent, nous avons

$$\|p - p_u\| = \frac{|p(u)|}{|r(u)|} \|r\| \leqslant \varepsilon \|d\|.$$

De plus comme $\|d\| = 1$, on a

$$\|p - p_u\| \leqslant \varepsilon.$$

Enfin puisque $p_u(u) = 0$, u appartient à $Z_\varepsilon(p)$. ∎

Ce théorème peut être facilement étendu au cas des systèmes polynomiaux.

Corollaire 9.1 (Stetter [185]). *L'ensemble des ε-pseudozéros complexes de $P = \{p_1, \ldots, p_k\}$, $k \in \mathbf{N}$ vérifie*

$$Z_\varepsilon(P) = \left\{ z \in \mathbf{C}^n : \frac{|p_l(z)|}{\|\mathbf{z_l}\|_*} \leqslant \varepsilon \text{ pour } l = 1, \ldots, k \right\},$$

où $\mathbf{z_l} := (\ldots, |z|^j, \ldots, j \in J_l)^T$.

Pour le prochain théorème, nous allons nous restreindre à la situation où P ainsi que tous les systèmes perturbés de $N_\varepsilon(P)$ sont 0-dimensionnels, c'est-à-dire si les solutions du système sont non vides et finies.

Théorème 9.2 (Stetter [185]). *Sous les hypothèse précédentes, chaque système $\widetilde{P} \in N_\varepsilon(P)$ a le même nombre de zéros (comptés avec leur multiplicités) dans chaque composante connexe de $Z_\varepsilon(P)$.*

Démonstration. Il s'agit de la preuve de [185, Thm. 3.5]. ■

9.3 Pseudozéros de polynômes à plusieurs indéterminées à coefficients réels

9.3.1 Pseudozéros complexes de polynômes réels

Le ε-voisinage réel de p est l'ensemble de tous les polynômes de $\mathcal{P}^n(\mathbf{R})$, suffisamment proche de p, c'est-à-dire,

$$N_\varepsilon^R(p) = \{\widetilde{p} \in \mathcal{P}^n(\mathbf{R}) : \|p - \widetilde{p}\| \leqslant \varepsilon\}.$$

On définit alors l'ensemble des ε-pseudozéros réels de p comme l'ensemble des zéros des polynômes du ε-voisinage réel de p. Formellement, on a

$$Z_\varepsilon^R(p) = \left\{z \in \mathbf{C}^n : \widetilde{p}(z) = 0 \text{ for } \widetilde{p} \in N_\varepsilon^R(p)\right\}.$$

Pour $\varepsilon = 0$, l'ensemble des pseudozéros $Z_0^R(p)$ n'est autre que l'ensemble des racines de p noté $Z(p)$.

Le théorème 9.3 suivant fournit une formule calculable des pseudozéros. Il est basé sur la preuve de [93, Theorem 5.1]. Pour $x, y \in \mathbf{R}^N$, on définit

$$d(x, \mathbf{R}y) = \inf_{\alpha \in \mathbf{R}} \|x - \alpha y\|_*,$$

la distance du point $x \in \mathbf{R}^N$ à la droite $\mathbf{R}y = \{\alpha y, \alpha \in \mathbf{R}\}$.

Théorème 9.3. *L'ensemble des ε-pseudozéros réels de $p = \sum_{j \in J} a_j z^j \in \mathcal{P}^n(\mathbf{R})$ satisfait*

$$Z_\varepsilon^R(p) = Z(p) \cup \left\{z \in \mathbf{C}^n \backslash Z(p) : h(z) := d(G_R(z), \mathbf{R}G_I(z)) \geqslant \frac{1}{\varepsilon}\right\},$$

où $G_R(z)$ et $G_I(z)$ sont respectivement les parties réelles et imaginaires de

$$G(z) = \frac{1}{p(z)}(\ldots, z^j, \ldots, j \in J)^T, \ z \in \mathbf{C}^n \backslash Z(p).$$

Démonstration. Soit $z \in Z_\varepsilon^R(p)$. Si $p(z) = 0$ alors $z \in Z(p)$ autrement il existe $q \in N_\varepsilon^R(p)$ tel que $q(z) = 0$. Dans ce cas, on a $p(z) = p(z) - q(z) = (p-q)^T \underline{z}$ où $\underline{z} = (\ldots, z^j, \ldots, j \in J)^T$. Ceci implique que $1 = (p-q)^T G(z)$. Ainsi nous avons $1 = (p-q)^T G_R(u) + i(p-q)^T G_I(u)$ et donc

$$\begin{cases} (p-q)^T G_R(u) = 1, \\ (p-q)^T G_I(u) = 0. \end{cases}$$

En conséquence, on a $\|p-q\| \|G_R(u) - \alpha G_I(u)\|_* \geqslant 1$ pour tout $\alpha \in \mathbf{R}$. En conclusion,

$$d(G_R(u), \mathbf{R}G_I(u)) \geqslant \frac{1}{\|p-q\|} \geqslant \frac{1}{\varepsilon}.$$

Réciproquement, soit $z \in Z(p) \cup \left\{ z \in \mathbf{C}^n \backslash Z(p) : d(G_R(z), \mathbf{R}G_I(z)) \geqslant \frac{1}{\varepsilon} \right\}$. Si z appartient à $Z(p)$alors il appartient aussi à $Z_\varepsilon^R(p)$. Sinon z vérifie $d(G_R(z), \mathbf{R}G_I(z)) \geqslant 1/\varepsilon$. D'après un théorème de dualité (voir [132, p.119]), il existe un vecteur $u \in \mathbf{R}^N$ satisfaisant $\|u\| = 1$ et

$$u^T G_R(z) = d(G_R(z), \mathbf{R}G_I(z)) \quad \text{et} \quad u^T G_I(z) = 0.$$

Considérons le polynôme à coefficients réels

$$q = p - \frac{u}{d(G_R(z), \mathbf{R}G_I(z))}.$$

Nous avons

$$q(z) = p(z) - \frac{u^T \underline{z}}{d(G_R(z), \mathbf{R}G_I(z))} = p(z) - \frac{p(z) u^T G(z)}{d(G_R(z), \mathbf{R}G_I(z))} = 0.$$

De plus on a $\|q-p\| = 1/d(G_R(z), \mathbf{R}G_I(z))$, si bien que $\|p-q\| \leqslant \varepsilon$. ■

Pour calculer l'ensemble des ε-pseudozéros réel $Z_\varepsilon^R(p)$, il nous suffit de savoir calculer la distance $d(G_R(z), \mathbf{R}G_I(z))$. On peut facilement la calculer pour la norme 2. En effet $\|\cdot\|_2$ la norme 2 et $\langle \cdot, \cdot \rangle$ le produit scalaire correspondant. Dans ce cas, nous avons

$$d(x, \mathbf{R}y) = \begin{cases} \sqrt{\|x\|_2^2 - \frac{\langle x,y \rangle^2}{\|y\|_2^2}} & \text{si } y \neq 0, \\ \|x\|_2 & \text{si } y = 0. \end{cases}$$

Pour la norme ∞, on peut montrer [114, Prop. 7.7.2] que

$$d(x, \mathbf{R}y) = \begin{cases} \min_{\substack{i=0:n \\ y_i \neq 0}} \|x - (x_i/y_i)y\|_1 & \text{si } y \neq 0, \\ \|x\|_1 & \text{si } y = 0. \end{cases}$$

Pour les autres normes p avec $p \neq 2, \infty$, il n'y a pas à notre connaissance de formule calculable pour $d(x, \mathbf{R}y)$.

La encore, le théorème précèdent peut être étendu facilement aux cas des systèmes polynomiaux.

Corollaire 9.2. *L'ensemble des ε-pseudozéros réels de $P = \{p_1, \ldots, p_k\}$, $k \in \mathbf{N}$ satisfait*

$$Z_\varepsilon^R(P) = \bigcap_{l=1}^{k} \left(Z(p_l) \cup \left\{ z \in \mathbf{C}^n \backslash Z(p_l) : h_l(z) := d(G_R^l(z), \mathbf{R}G_I^l(z)) \geqslant \frac{1}{\varepsilon} \right\} \right),$$

où $G_R^l(z)$ et $G_I^l(z)$ sont respectivement les parties réelle et imaginaires de

$$G^l(z) = \frac{1}{p_l(z)}(\ldots, z^j, \ldots, j \in J_l)^T, \ z \in \mathbf{C}^n \backslash Z(p).$$

Comme nous venons de le voir, l'ensemble des pseudozéros réels est intimement relié à la fonction d. Cette fonction peut avoir des discontinuité. C'est le sujet du lemme suivant.

Lemme 9.1 (Hinrichsen et Kelb [89])**.** *La fonction*

$$d : \mathbf{R}^{n+1} \times \mathbf{R}^{n+1} \to \mathbf{R}_+, \quad (x, y) \mapsto d(x, \mathbf{R}y)$$

est continue en tous les points (x, y) tel que $y \neq 0$ où $x = 0$ et discontinue en tous les points $(x, 0) \in \mathbf{R}^{n+1} \times \mathbf{R}^{n+1}$, $x \neq 0$.

Ce lemme établit que des discontinuités surviennent quand le vecteur y s'annule. Dans notre cas, les discontinuités surviennent que $G_I(z) = 0$ où $G_I(z)$ est la partie imaginaire de

$$G(z) = \frac{1}{p(z)}(1, z, \ldots, z^n)^T.$$

On en déduit alors que G_I s'annule pour $z \in \mathbf{R}$, c'est-à-dire sur l'axe réel. Ceci explique pourquoi les commandes MATLAB `contour` et `meshc` peuvent donner de mauvais résultats le long de l'axe réel. Bien entendu, si aucun des zéros de notre polynôme n'est réel, alors on obtient de très bons résultats pour le tracé.

9.3.2 Pseudozéros réels de polynômes réels

Dans les sous-sections précédentes, nous nous sommes intéressés aux zéros complexes d'un système de polynômes réels. Il peut arriver que nous soyons intéressés seulement par les zéros réels du système. Autrement dit, étant donné un polynôme $p \in \mathcal{P}^n(\mathbf{R})$, nous voulons calculer $Z_\varepsilon^R(p) \cap \mathbf{R}^n$. Le théorème suivant donne une formule explicite pour calculer cet ensemble.

Théorème 9.4. *L'intersection entre l'ensemble des ε-pseudozéros complexes de $p = \sum_{j \in J} a_j z^j \in \mathcal{P}^n(\mathbf{R})$ et $\mathbf{R}^n$ vérifie*

$$Z_\varepsilon^R(p) \cap \mathbf{R}^n = \left\{ z \in \mathbf{R}^n : g(z) := \frac{|p(z)|}{\|\mathbf{z}\|_*} \leqslant \varepsilon \right\},$$

où $\mathbf{z} := (\ldots, |z|^j, \ldots, j \in J)^T$.

Démonstration. Si $z \in Z_\varepsilon^R(p) \cap \mathbf{R}^n$ alors il existe $\widetilde{p} \in \mathcal{P}^n(\mathbf{R})$ tel que $\widetilde{p}(z) = 0$ et $\|p - \widetilde{p}\| \leqslant \varepsilon$. D'après l'inégalité de Hölder généralisée $|x^T y| \leqslant \|x\| \|y\|_*$, on a

$$|p(z)| = |p(z) - \widetilde{p}(z)| = \left| \sum_{j \in J} (p_j - \widetilde{p}_j) z^j \right| \leqslant \|p - \widetilde{p}\| \|\mathbf{z}\|_*.$$

On en déduit que $|p(z)| \leqslant \varepsilon \|\mathbf{z}\|_*$.
Réciproquement, soit $u \in \mathbf{R}$ tel que $|p(u)| \leqslant \varepsilon \|\mathbf{u}\|$ avec $\mathbf{u} := (\ldots, |u|^j, \ldots, j \in J)$. Le vecteur dual $d \in \mathbf{R}^N$ de $\mathbf{u}$ satisfait $d^T \mathbf{u} = \|\mathbf{u}\|_*$ et $\|d\| = 1$ Introduisons les deux polynômes r et p_u par

$$\begin{aligned} r(z) &= \sum_{j \in J} d_j z^j, \\ p_u(z) &= p(z) - \frac{p(u)}{r(u)} r(z). \end{aligned}$$

Il est clair que $r(u) = d^T \mathbf{u} = \|\mathbf{u}\|_*$. Par conséquent, nous avons

$$\|p - p_u\| = \frac{|p(u)|}{|r(u)|} \|r\| \leqslant \varepsilon \|d\|.$$

Puisque $\|d\| = 1$, on a

$$\|p - p_u\| \leqslant \varepsilon.$$

De plus comme $p_u(u) = 0$, on en déduit que u appartient à $Z_\varepsilon^R(p) \cap \mathbf{R}^n$. ■

Ce théorème s'étend facilement aux systèmes polynomiaux.

Corollaire 9.3. *L'intersection entre l'ensemble des ε-pseudozéros de $P = \{p_1, \ldots, p_k\}$, $k \in \mathbf{N}$ et $\mathbf{R}^n$ vérifie*

$$Z_\varepsilon^R(P) \cap \mathbf{R}^n = \left\{ z \in \mathbf{R}^n : \frac{|p_l(z)|}{\|\mathbf{z_l}\|_*} \leqslant \varepsilon \text{ for } l = 1, \ldots, k \right\},$$

où $\mathbf{z_l} := (\ldots, |z|^j, \ldots, j \in J_l)^T$.

9.4 Visualisation de l'ensemble des pseudozéros

La description de $Z_\varepsilon(P)$ et $Z_\varepsilon^R(P)$ donnée par le théorème 9.1 et le théorème 9.3 (et aussi par le corollaire 9.1 et le corollaire 9.2) nous permet de calculer, tracer et visualiser l'ensemble des pseudozéros d'un polynôme ou d'un système de polynômes à plusieurs indéterminées. L'ensemble des pseudozéros est un sous-ensemble de $\mathbf{C}^n$ que l'on ne peut représenter qu'à travers ses projections sur un ensemble de dimension plus petite qui est très souvent $\mathbf{C}$.

Pour ce faire, nous avons écrit un package MATLAB pour calculer et visualiser ces projections (voir la section 9.6). Ce package nécessite la Symbolic Math Toolbox (et la

Extended Symbolic Math Toolbox) qui est une interface entre MATLAB et le noyau de MAPLE.

Pour un $v \in \mathbf{C}^n$ donné, soit $Z_\varepsilon(P, j, v)$ la projection de $Z_\varepsilon(P)$ sur le z_j-espace au voisinage de v. On déduit pour $P = \{p_1, \ldots, p_k\}$ que

$$Z_\varepsilon(P, j, v) = \left\{ z \in \mathbf{C}^n : z_i = v_i \text{ pour } i \neq j, \text{ et } \max_{l=1,\ldots,k} \frac{|p_l(z)|}{\|\mathbf{z_l}\|_*} \leqslant \varepsilon \right\},$$

avec $\mathbf{z_l} := (\ldots, |z|^j, \ldots, j \in J_l)^T$. Une façon de visualiser $Z_\varepsilon(P, j, v)$ est de calculer la projection de

$$\mathrm{ps}(z) := \log_{10}\left(\max_{l=1,\ldots,k} \frac{|p_l(z)|}{\|\mathbf{z_l}\|_*} \right)$$

sur une grille de points autour de v dans le z_j-espace. De la même façon, on définit, pour un $v \in \mathbf{C}^n$ donné, $Z_\varepsilon^R(P, j, v)$ la projection de $Z_\varepsilon^R(P)$ sur le z_j-espace au voisinage de v. On en déduit alors pour $P = \{p_1, \ldots, p_k\}$, que

$$Z_\varepsilon^R(P, j, v) = \left\{ z \in \mathbf{C}^n : z_i = v_i \text{ pour } i \neq j, \text{ et } \max_{l=1,\ldots,k} d(G_R^l(z), \mathbf{R}G_I^l(z))^{-1} \leqslant \varepsilon \right\},$$

où $G_R^l(z)$ et $G_I^l(z)$ sont respectivement les parties réelles et imaginaires de

$$G^l(z) = \frac{1}{p_l(z)}(\ldots, z^j, \ldots, j \in J_l)^T, \ z \in \mathbf{C}^n \backslash Z(p).$$

Une façon de visualiser $Z_\varepsilon^R(P, j, v)$ est encore de calculer la projection de

$$\mathrm{ps}^R(z) := \log_{10}\left(\max_{l=1,\ldots,k} d(G_R^l(z), \mathbf{R}G_I^l(z))^{-1} \right)$$

sur une grille de points autour de v dans le z_j-espace. Examinons l'exemple suivant tiré de [98] (voir la figure 9.1) en utilisant la norme 2 : l'intersection de deux boules en $(2, 2)$,

$$P_1 = \begin{cases} p_1 = (z_1 - 1)^2 + (z_2 - 2)^2 - 1, \\ p_2 = (z_1 - 3)^2 + (z_2 - 2)^2 - 1. \end{cases}$$

On peut ne s'intéresser qu'aux pseudozéros réels de systèmes polynomiaux. Dans ce cas on ne trace que $\mathbf{R}^n \cap Z_\varepsilon^R(P)$. C'est ce que nous faisons dans l'exemple suivant dans la figure 9.2 (en utilisant toujours la norme 2),

$$P_2 = \begin{cases} p_1 = z_1^2 + z_2^2 - 1, \\ p_2 = 25 z_1 z_2 - 12. \end{cases}$$

Dans cette figure, nous avons calculer la fonction

$$g(x, y) = \max_{l=1,2} \frac{p_l(x, y)}{\|\mathbf{z_l}\|_*},$$

avec $\mathbf{z_l} := (\ldots, |x + iy|^j, \ldots, j \in J_l)^T$.

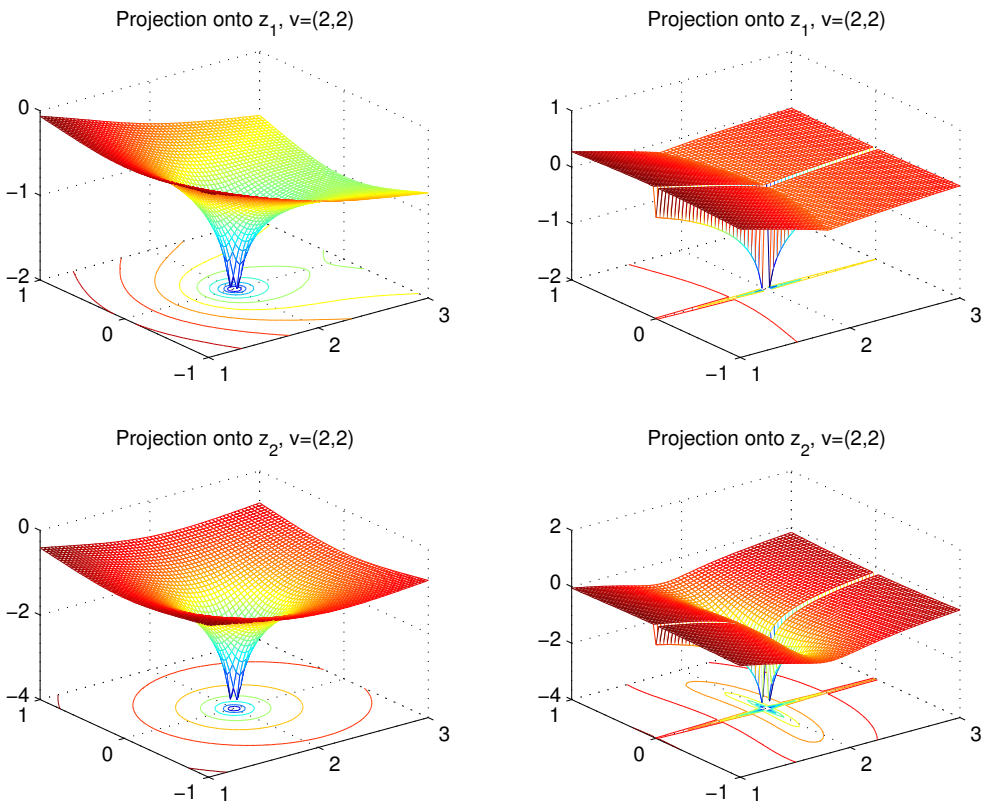


FIG. 9.1 – Projections de l'ensemble des pseudozéros complexes (à gauche) et de l'ensemble des pseudozéros réels (à droite) de P_1

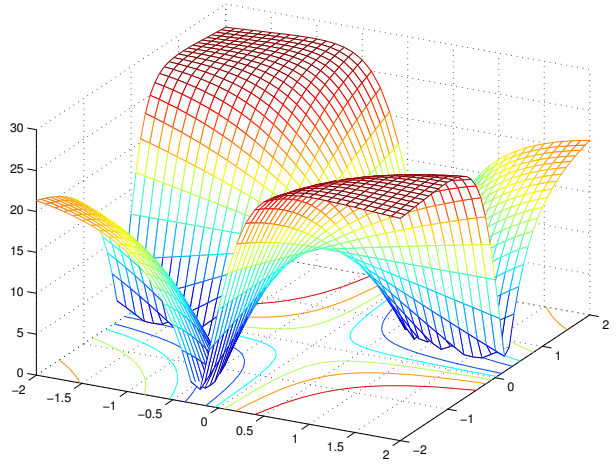


FIG. 9.2 – Projection de l'ensemble des pseudozéros de P_2

9.5 Conclusion

Dans ce chapitre, nous avons donné une formule calculable pour l'ensemble des pseudozéros réels d'un polynôme et d'un système de polynômes à plusieurs indéterminées. Nous avons montré que cela permet de mesurer la sensibilité des zéros de systèmes polynomiaux. L'ensemble des pseudozéros offre donc un outil potentiellement utile dans les problèmes de calcul de zéros de systèmes polynomiaux.

9.6 Code MATLAB des programmes

```
function [] = pseudo(polys,indets,proj,coord,xaxis,yaxis)
% polys   : system of polynomials
% indets  : variables
% proj    : variable where we project
% coord   : coordinate of the point near we project
% xaxis   : coordinate for the x-axis
% yaxis   : coordinate for the y-axis
% example :
% pseudo({'(x-1)^2+(y-2)^2-1','(x-3)^2+(y-2)^2-1'}, ...
%         {'x','y'},'x',[2 2],1:0.02:3,-1:0.02:1)

% load of a maple function that
% give the list of the monomial
% of a polynomial
procread('monomial.maple');

% number of variable in the system
nbindets = length(indets);

% put the variables as symbolic variables
for k = 1:nbindets
   syms(indets{k});
end

% number of polynomials in the system
nbpoly = length(polys);

monomials = {};
for k=1:nbpoly
   monomials{k} = maple('monomial',polys{k});
end

% substitute a value to variables which do not change
```

```
ind = 0; % index of the variable that moves
for k = 1:nbindets
   if (indets{k} ~= proj);
      for j=1:nbpoly
         polys{j} = simplify(subs(polys{k},indets{k},coord(k)));
         dual{j} = simplify(subs(monomials{j},indets{k},coord(k)));
      end
   else
      ind = k;
   end
end

x= xaxis;
y= yaxis;

% Define a grid
[X,Y] = meshgrid(x,y);

% size of the grid
[r,s] = size(X);

% Transform (x,y) of the grid in complex numbers as z=x+iy
Z = X + i.*Y;

for l=1:r
   for j=1:s
      tab = [];
      for k=1:nbpoly
         % compute the function that check the pseudozero set
         num = subs(polys{k},indets{ind},Z(l,j));
         denum = norm(subs(dual{k},indets{ind},Z(l,j)),2);
         tab = [log10(abs(num)/abs(denum)) tab];
      end
      Res(l,j) = max(tab);
   end
end

% draw the result
meshc(x,y,Res);
```

Dans le programme précédent, nous avons utilisé la fonction MAPLE suivante.

```
monomial := proc(poly)
```

```
local listmono,mono,nbmono,k,p;

listmono := [op(expand(poly))];
nbmono := nops(listmono);

for k from 1 to nbmono do
   mono := listmono[k];
   mono := simplify(abs(mono/coeffs(mono)));
   listmono[k] := mono;
od;
return(listmono);
end;
```

Chapitre 10

Conditionnement structuré de racines de polynômes

Ce chapitre est le pendant du chapitre 2 pour le problème du calcul des racines de polynômes. Les outils et les techniques utilisés sont très similaires.

Dans ce chapitre, nous nous intéressons au conditionnement du problème de la détermination d'une racine simple d'un polynôme donné. Le conditionnement d'une racine simple a été étudié dans de nombreux articles parmi lesquels [198, 61, 100, 48, 36] pour des polynômes à coefficients complexes. Cela signifie que l'on autorise des perturbations complexes sur les coefficients complexes du polynôme. Cependant, pour un polynôme à coefficients réels, il semble encore plus naturel de n'autoriser seulement que des perturbations réels des coefficients réels. C'est par exemple le cas quand les coefficients du polynôme décrivent des valeurs physiques (voir le chapitre 8 pour les polynômes d'intervalles). Une autre motivation pour contraindre le conditionnement à décrire des perturbations réelles vient du calcul en précision finie. En effet, les erreurs d'arrondi d'un calcul sur des nombres réels en précision finie sont des nombres réels. Nous pouvons donc définir un conditionnement réel ne prennant en compte que les perturbations réelles d'un polynôme réel. La question de savoir si ce conditionnement réel est différent (et si oui dans quelle mesure) du conditionnement classique (complexe) se pose alors naturellement.

La principale contribution de ce chapitre est la définition et l'obtention d'une formule calculable pour le conditionnement et l'erreur inverse réelle pour le calcul de racines de polynômes. Nous montrons ainsi qu'il y peu de différence entre le conditionnement réel et le conditionnement classique. Par contre, la différence peut être importante entre l'erreur inverse réelle et l'erreur inverse classique.

Le reste du chapitre est organisé de la manière suivante. Dans la section 10.1, nous considérons le conditionnement d'une racine simple d'un polynôme. Nous étudions d'abord le conditionnement d'une racine simple complexe d'un polynôme à coefficients complexes en considérant des perturbations complexes. Nous étudions ensuite le conditionnement d'une racine complexe d'un polynôme à coefficients réels en ne considérant que des per-

turbations réelles des coefficients. Dans la section 10.2, nous nous intéressons à l'erreur inverse pour le problème du calcul de racines de polynômes. Là encore, nous considérons tout d'abord des polynômes à coefficients complexes et ensuite des polynômes à coefficients réels.

10.1 Conditionnement de racines de polynômes

Dans cette section, nous considérons le conditionnement d'une racine simple d'un polynôme donné. Nous commençons d'abord par le cas d'un polynôme à coefficients complexes. Nous nous restreignons ensuite au cas d'un polynôme à coefficients réels. Enfin, nous étudions le conditionnement des parties réelles et imaginaires d'une racine simple d'un polynôme réel.

10.1.1 Conditionnement de polynômes à coefficients complexes

Pour $n \geqslant 1$, on définit par $\mathcal{P}_n(\mathbf{C})$ l'espace des polynômes à coefficients complexes de degré au plus n. Soit $p \in \mathcal{P}_n(\mathbf{C})$ un polynôme de degré n donné par

$$p(z) = \sum_{i=0}^{n} p_i z^i.$$

Nous représenterons souvent p par le vecteur de ses coefficients $(p_0, p_1, \ldots, p_n)^T$. Nous identifierons aussi la norme $\|\cdot\|$ sur $\mathcal{P}_n(\mathbf{C})$ à la norme 2 sur $\mathbf{C}^{n+1}$ du vecteur correspondant. Dans ce chapitre nous travaillerons uniquement avec la norme 2 notée $\|\cdot\|$. Cela signifie que

$$\|p\| = \left(\sum_{i=0}^{n} |p_i|^2 \right)^{1/2}. \tag{10.1}$$

Définition 10.1. Soit $p \in \mathcal{P}_n(\mathbf{C})$ un polynôme de degré n et z_0 une racine simple de p. Le conditionnement (absolu) en z_0 est

$$\kappa^{\mathbf{C}}(z_0, p) := \lim_{\varepsilon \to 0} \sup \left\{ \frac{|\delta z_0|}{\|\delta p\|} : \delta p \in \mathcal{P}_n(\mathbf{C}),\ \|\delta p\| = \varepsilon \right\}.$$

Dans la définition précédente, δz_0 représente la variation de la racine z_0 quand on perturbe le polynôme p par un polynôme δp. Cela signifie juste que $z_0 + \delta z_0$ est une racine de $p + \delta p$. Le théorème suivant fournit une expression calculable pour le conditionnement.

Théorème 10.1 (Chatelin et Frayssé [36])**.** *Soient $p \in \mathcal{P}_n(\mathbf{C})$ un polynôme de degré n et z_0 une racine simple de p. Le conditionnement de z_0 est donné par*

$$\kappa^{\mathbf{C}}(z_0, p) = \frac{\|\underline{z_0}\|}{|p'(z_0)|},$$

où $\underline{z_0} := (1, z_0, z_0^2, \ldots, z_0^n)^T$.

Démonstration. Soit $p(z) = \sum_{i=0}^{n} p_i z^i$ et considérons l'application

$$\varphi : \left\{ \begin{array}{rcl} \mathcal{P}_n(\mathbf{C}) \simeq \mathbf{C}^{n+1} & \longrightarrow & \mathbf{C}, \\ (p_0, \ldots, p_n)^T & \longmapsto & z \text{ tel que } p(z) = 0, \ z \text{ simple.} \end{array} \right.$$

De la définition de φ, on déduit que $p(\varphi(p)) = 0$. La règle de dérivation des fonctions composées montre que

$$\frac{\partial \varphi}{\partial p_i}(p) p'(z) + z^i = 0, \quad i = 0 : n.$$

Cela entraîne que $\varphi'(p) = -\frac{1}{p'(z)} \underline{z}^T$. En utilisant le développement en série de Taylor, on obtient

$$z_0 + \delta z_0 = z_0 - \frac{1}{p'(z_0)} \underline{z_0}^T \delta p + \mathcal{O}(\|\delta p\|^2). \tag{10.2}$$

De la définition 10.1, on déduit

$$\kappa^{\mathbf{C}}(z_0, p) = \frac{1}{|p'(z_0)|} \sup \left\{ |\underline{z_0}^T \delta p|, \ \|\delta p\| = 1 \right\}.$$

Il est facile de montrer que $\kappa^{\mathbf{C}}(z_0, p) \leqslant \frac{1}{|p'(z_0)|} \|\underline{z_0}\|$. Notons $z_0 = |z_0| e^{i\theta}$ and $\delta p = (|z_0|^k e^{-ik\theta})_{k=0:n}$. Nous avons $\underline{z_0}^T \delta p = \|\underline{z_0}\|$. Cela termine la preuve. ■

10.1.2 Conditionnement de polynômes à coefficients réels

Dans certains cas, le polynôme p est à coefficients réels et nous savons que les perturbations éventuelles sur les coefficients resteront réels. La question de savoir si le conditionnement réel est différent (et si oui dans quelle mesure) se pose alors naturellement.

Les notations sont similaires au cas complexe. Pour $n \geqslant 1$, on note $\mathcal{P}_n(\mathbf{R})$ l'ensemble des polynômes à coefficients réels de degré au plus n. Un polynôme p de $\mathcal{P}_n(\mathbf{R})$ est représenté par le vecteur $(p_0, p_1, \ldots, p_n)^T$ de ses coefficients, *i.e.* $p(z) = \sum_{i=0}^{n} p_i z^i$. Nous identifions toujours la norme $\|\cdot\|$ sur $\mathcal{P}_n(\mathbf{R})$ à la norme 2 sur $\mathbf{R}^{n+1}$ du vecteur correspondant (relation (10.1)).

Définition 10.2. Soient $p \in \mathcal{P}_n(\mathbf{R})$ un polynôme de degré n et z_0 une racine simple de p. Le conditionnement réel de z_0 est défini par

$$\kappa^{\mathbf{R}}(z_0, p) := \lim_{\varepsilon \to 0} \sup \left\{ \frac{|\delta z_0|}{\|\delta p\|} : \delta p \in \mathcal{P}_n(\mathbf{R}), \ \|\delta p\| = \varepsilon \right\}.$$

Le théorème suivant fournit une expression calculable pour le conditionnement réel. Un résultat similaire a été obtenu par Hough [100]. La différence vient du fait qu'il utilise des polynômes unitaires et qu'il définit les perturbations sur les racines au lieu des coefficients.

Théorème 10.2. *Soient $p \in \mathcal{P}_n(\mathbf{R})$ un polynôme de degré n et z_0 une racine simple de p. Le conditionnement réel de z_0 est donné par*

$$\kappa^{\mathbf{R}}(z_0,p) = \frac{1}{\sqrt{2}|p'(z_0)|}\sqrt{(\|u\|^2+\|v\|^2+((\|u\|^2-\|v\|^2)^2-4(u^Tv)^2)^{1/2})},$$

où $\underline{z_0} := (1, z_0, z_0^2, \ldots, z_0^n)^T$ et u, v sont respectivement les parties réelles et imaginaires de $\underline{z_0}$.

Démonstration. De la preuve du théorème 10.1, on déduit que

$$\kappa^{\mathbf{R}}(z_0,p) = \frac{1}{|p'(z_0)|}\sup\left\{|\underline{z_0}^T\delta p|,\ \delta p \in \mathcal{P}_n(\mathbf{R}),\ \|\delta p\|=1\right\}.$$

En d'autres termes, nous avons

$$\kappa^{\mathbf{R}}(z_0,p)^2 = \frac{1}{|p'(z_0)|^2}\sup\left\{|\underline{z_0}^T\delta p|^2,\ \delta p \in \mathcal{P}_n(\mathbf{R}),\ \|\delta p\|=1\right\}.$$

Écrivons $\underline{z_0} = u + iv$, où u et v sont respectivement les parties réelles et imaginaires de $\underline{z_0}$. Avec ces notations, on a

$$|\underline{z_0}^T\delta p|^2 = \delta p^T(uu^T+vv^T)\delta p.$$

En appliquant le théorème du quotient de Rayleigh (voir [99, p.176]) on trouve

$$\sup\left\{|\underline{z_0}^T\delta p|^2,\ \delta p \in \mathcal{P}_n(\mathbf{R}),\ \|\delta p\|=1\right\} = \lambda_{\max}(uu^T+vv^T),$$

où $\lambda_{\max}(A)$ dénote la plus grande valeur propre de la matrice A. La matrice uu^T+vv^T est de rang 2 et son image est l'ensemble $\mathrm{Vect}(u,v)$. Soit λ une valeur propre de uu^T+vv^T et $au+bv$, $a,b \in \mathbf{R}$ un vecteur propre. Il s'en suit que

$$(uu^T+vv^T)(au+bv) = \lambda(au+bv).$$

On peut réécrire cela sous la forme

$$\begin{pmatrix} u^Tu & u^Tv \\ v^Tu & v^Tv \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = \lambda\begin{pmatrix} a \\ b \end{pmatrix}.$$

Les valeurs propres de cette matrice sont les racines de son polynôme caractéristique qui est

$$P(X) = X^2 - (u^Tu+v^Tv)X + (u^Tu)(v^Tv) - (u^Tv)^2.$$

Les deux racines de ce polynôme sont

$$\begin{cases} X_1 = \frac{1}{2}(\|u\|^2+\|v\|^2+((\|u\|^2-\|v\|^2)^2-4(u^Tv)^2)^{1/2}), \\ X_2 = \frac{1}{2}(\|u\|^2+\|v\|^2-((\|u\|^2-\|v\|^2)^2-4(u^Tv)^2)^{1/2}). \end{cases}$$

En conséquence

$$\lambda_{\max}(uu^T+vv^T) = \frac{1}{2}(\|u\|^2+\|v\|^2+((\|u\|^2-\|v\|^2)^2-4(u^Tv)^2)^{1/2}),$$

et le résultat suit de ce que $\kappa^{\mathbf{R}}(z_0,p) = \frac{1}{|p'(z_0)|}\sqrt{\lambda_{\max}(uu^T+vv^T)}$. ∎

Soit $p \in \mathcal{P}_n(\mathbf{R})$ un polynôme à coefficients réels. Du fait que

$$\begin{aligned}\kappa^{\mathbf{R}}(z_0,p) &= \frac{1}{\sqrt{2}|p'(z_0)|}\sqrt{(\|u\|^2+\|v\|^2+((\|u\|^2-\|v\|^2)^2-4(u^Tv)^2)^{1/2})},\\ \kappa^{\mathbf{C}}(z_0,p) &= \frac{\sqrt{\|u\|^2+\|v\|^2}}{|p'(z_0)|},\end{aligned}$$

on déduit

$$\kappa^{\mathbf{C}}(z_0,p)/\sqrt{2} \leqslant \kappa^{\mathbf{R}}(z_0,p) \leqslant \kappa^{\mathbf{C}}(z_0,p).$$

Prenons l'exemple $p(z) = z^3+z$ et $z_0 = i$. Nous avons $\kappa^{\mathbf{R}}(i,p) = 1/\sqrt{2}$ et $\kappa^{\mathbf{C}}(i,p) = 1$. De la même façon, si nous choisissons $p(z) = z-1$ et $z_0 = 1$ alors on a $\kappa^{\mathbf{R}}(1,p) = \kappa^{\mathbf{C}}(1,p) = \sqrt{2}$. Cela signifie que les bornes présentées ci-dessus sont optimales.

10.1.3 Conditionnement de parties réelles et imaginaires

Si la perturbation δp est réelle et si nous notons $\underline{z_0} = u+iv$, $p'(z_0) = x+iy$ alors l'équation (10.2) peut s'écrire sous la forme

$$\operatorname{Re}(\delta z_0) = -\frac{(xu^T+yv^T)\delta p}{|p'(z_0)|^2} + \mathcal{O}(\|\delta p\|^2), \tag{10.3}$$

$$\operatorname{Im}(\delta z_0) = -\frac{(xv^T-yu^T)\delta p}{|p'(z_0)|^2} + \mathcal{O}(\|\delta p\|^2). \tag{10.4}$$

En suivant [168], on peut définir un conditionnement pour les parties réelles et imaginaires de la racine z_0 de la manière suivante

$$\begin{aligned}\kappa^{\mathbf{R}}_{\Re}(z_0,p) &:= \lim_{\varepsilon\to 0}\sup\left\{\frac{|\operatorname{Re}(\delta z_0)|}{\|\delta p\|} : \delta p \in \mathcal{P}_n(\mathbf{R}),\ \|\delta p\| = \varepsilon\right\},\\ \kappa^{\mathbf{R}}_{\Im}(z_0,p) &:= \lim_{\varepsilon\to 0}\sup\left\{\frac{|\operatorname{Im}(\delta z_0)|}{\|\delta p\|} : \delta p \in \mathcal{P}_n(\mathbf{R}),\ \|\delta p\| = \varepsilon\right\}.\end{aligned}$$

À l'aide des équations (10.3) and (10.4), on obtient

$$\begin{aligned}\kappa^{\mathbf{R}}_{\Re}(z_0,p) &= \frac{1}{|p'(z_0)|^2}\sup\left\{|(xu^T+yv^T)\delta p| : \delta p \in \mathcal{P}_n(\mathbf{R}),\ \|\delta p\| = 1\right\},\\ \kappa^{\mathbf{R}}_{\Im}(z_0,p) &= \frac{1}{|p'(z_0)|^2}\sup\left\{|(xv^T-yu^T)\delta p| : \delta p \in \mathcal{P}_n(\mathbf{R}),\ \|\delta p\| = 1\right\}.\end{aligned}$$

Un calcul simple montre alors que

$$\begin{aligned}\kappa^{\mathbf{R}}_{\Re}(z_0,p) &= \frac{\|(xu^T+yv^T)\|}{|p'(z_0)|^2},\\ \kappa^{\mathbf{R}}_{\Im}(z_0,p) &= \frac{\|(xv^T-yu^T)\|}{|p'(z_0)|^2}.\end{aligned}$$

On remarque que $\kappa^{\mathbf{R}}_{\Re}(z_0,p)$ et $\kappa^{\mathbf{R}}_{\Im}(z_0,p)$ peuvent avoir des ordres de grandeur très différents. En effet, si $x = y$ et $u = v$ alors on a $\kappa^{\mathbf{R}}_{\Im}(z_0,p) = 0$ et $\kappa^{\mathbf{R}}_{\Re}(z_0,p) = \frac{\|2xu^T\|}{|p'(z_0)|^2}$. On en déduit que la partie imaginaire d'une racine peut être bien conditionnée alors que la partie réelle est elle très mal conditionnée.

10.2 Erreur inverse de racines de polynômes

Dans cette section, nous nous intéressons à l'erreur inverse d'une racine d'un polynôme. Comme dans les sections précédentes, nous regardons d'abord le problème avec des polynômes à coefficients complexes et nous autorisons des perturbations complexes de ses coefficients. Nous nous restreignons ensuite aux polynômes à coefficients réels et nous n'autorisons alors que des perturbations réelles de ses coefficients. À la fin de la section, nous établissons un lien entre l'ensemble des pseudozéros et l'erreur inverse.

10.2.1 Erreur inverse de racines de polynômes à coefficients complexes

Définition 10.3. Soient $p \in \mathcal{P}_n(\mathbf{C})$ un polynôme de degré n et z une racine approchée de p. L'erreur inverse en z est définie comme

$$\eta^{\mathbf{C}}(z) := \min\{\|\delta p\| : \delta p \in \mathcal{P}_n(\mathbf{C}),\ (p+\delta p)(z) = 0\}.$$

Théorème 10.3 (Chatelin et Frayssé [36])**.** *Soient $p \in \mathcal{P}_n(\mathbf{C})$ un polynôme de degré n et z une racine approchée de p. L'erreur inverse de z est donnée par*

$$\eta^{\mathbf{C}}(z) = \frac{|p(z)|}{\|\underline{z}\|},$$

où $\underline{z} := (1, z, z^2, \ldots, z^n)^T$

Démonstration. Nous supposons que $p(z) \neq 0$, sinon on choisit $\delta p = 0$ et alors $\eta^{\mathbf{C}}(z) = 0$. Soit $\delta p \in \mathcal{P}_n(\mathbf{C})$ tel que $(p+\delta p)(z) = 0$. On a alors $p(z) = p(z) - (p+\delta p)(z) = -\delta p^T \underline{z}$ avec $\underline{z} = (1, z, z^2, \ldots, z^n)$. On en déduit que $|p(z)| \leqslant \|\delta\|\|\underline{z}\|$. Ainsi nous avons

$$\eta^{\mathbf{C}}(z) \geqslant \frac{|p(z)|}{\|\underline{z}\|}.$$

Notons $z = |z|e^{i\theta}$. La borne inférieure est atteinte par $\delta p = (|z|^k e^{-ik\theta})_{k=0:n}$. ■

10.2.2 Erreur inverse de racines de polynômes à coefficients réels

Définition 10.4. Soient $p \in \mathcal{P}_n(\mathbf{R})$ un polynôme de degré n et z une racine approchée de p. L'erreur inverse réelle en z est

$$\eta^{\mathbf{R}}(z) := \min\{\|\delta p\| : \delta p \in \mathcal{P}_n(\mathbf{R}),\ (p+\delta p)(z) = 0\}.$$

Pour obtenir une expression calculable de cette erreur inverse, nous allons utiliser des arguments issus de [93]. Nous supposons que $p(z) \neq 0$, sinon on choisit $\delta p = 0$ et $\eta^{\mathbf{R}}(z) = 0$. Soit $\delta p \in \mathcal{P}_n(\mathbf{R})$ tel que $(p+\delta p)(z) = 0$. On a alors $p(z) = p(z) - (p+\delta p)(z) = -\delta p^T \underline{z}$ avec $\underline{z} = (1, z, z^2, \ldots, z^n)^T$. On en déduit que $1 = -\delta p^T G(z)$ où $G(z) = \frac{1}{p(z)}(1, z, \ldots, z^n)^T$. Posons $G(z) = G_R(z) + iG_I(z)$ où $G_R(z)$, $G_I(z)$ représentent respectivement les parties

réelles et imaginaires de $G(z)$. On en déduit $1 = -\delta p^T G_R(z) - i\delta p^T G_I(z)$. Par conséquent, nous avons

$$\begin{cases} \delta p^T G_R(z) = -1, \\ \delta p^T G_I(z) = 0. \end{cases}$$

Ainsi $\|\delta p\| \|G_R(z) - \alpha G_I(z)\| \geqslant 1$, pour tout $\alpha \in \mathbf{R}$. Étant donnés $x, y \in \mathbf{R}^{n+1}$, on définit par

$$d(x, \mathbf{R}y) = \inf_{\alpha \in \mathbf{R}} \|x - \alpha y\|,$$

le distance du point $x \in \mathbf{R}^{n+1}$ à la droite $\mathbf{R}y = \{\alpha y, \alpha \in \mathbf{R}\}$. On en déduit alors que $\|\delta p\| \geqslant \frac{1}{d(G_R(z), \mathbf{R}G_I(z))}$. D'après un théorème de dualité (voir [132, p.119]), il existe un vecteur $u \in \mathbf{R}^{n+1}$, $\|u\| = 1$, satisfaisant

$$u^T G_R(z) = d(G_R(z), \mathbf{R}G_I(z)) \quad \text{et} \quad u^T G_I(z) = 0.$$

Considérons maintenant le polynôme dont les coefficients sont donnés par

$$\delta p := -\frac{u}{d(G_R(z), \mathbf{R}G_I(z))}.$$

Théorème 10.4. *L'erreur inverse réelle en z est donnée par*

$$\eta^{\mathbf{R}}(z) = \frac{1}{d(G_R(z), \mathbf{R}G_I(z))},$$

Démonstration. Nous avons

$$(p + \delta p)(z) = p(z) - \frac{u^T \underline{z}}{d(G_R(z), \mathbf{R}G_I(z))} = p(z) - \frac{p(z) u^T G(z)}{d(G_R(z), \mathbf{R}G_I(z))} = 0$$

et

$$\|\delta p\| = \left\| \frac{u}{d(G_R(z), \mathbf{R}G_I(z))} \right\| = \frac{1}{d(G_R(z), \mathbf{R}G_I(z))}.$$

On en déduit que $p + \delta p$ est le polynôme réel le plus proche de p ayant z comme racine. ■

La principale difficulté est le calcul de $d(G_R(z), \mathbf{R}G_I(z))$. Si on note $\langle \cdot, \cdot \rangle$ le produit scalaire associé à la norme $\| \cdot \|$, on a

$$d(x, \mathbf{R}y) = \begin{cases} \sqrt{\|x\|_2^2 - \frac{\langle x, y \rangle^2}{\|y\|_2^2}} & \text{si } y \neq 0, \\ \|x\|_2 & \text{si } y = 0. \end{cases}$$

De la relation $d(G_R(z), \mathbf{R}G_I(z)) \leqslant \|G(z)\|$, on déduit que $\eta^{\mathbf{C}}(z) \leqslant \eta^{\mathbf{R}}(z)$ pour $p \in \mathcal{P}_n(\mathbf{R})$.

Remarque. On peut se demander si $\eta^{\mathbf{C}}(z) \approx \eta^{\mathbf{R}}(z)$. En fait, $\eta^{\mathbf{C}}(z)$ peut être très différent de $\eta^{\mathbf{R}}(z)$. Si par exemple $p(z) = z$, un calcul direct montre que $\eta^{\mathbf{R}}(z) = 1$ alors que $\eta^{\mathbf{C}}(z) = (1 + 1/|z|^2)^{-1/2}$ et donc $\eta^{\mathbf{C}}(z) \to 0$ quand $z \to 0$.

10.2.3 Erreur inverse et ensemble des pseudozéros

Le théorème 10.5 ci-dessous donne une relation entre l'erreur inverse et l'ensemble des pseudozéros.

Théorème 10.5. *L'ensemble des ε-pseudozéros de p satisfait*

$$Z_\varepsilon(p) = \left\{ z \in \mathbf{C} : \eta^{\mathbf{C}}(z) \leqslant \varepsilon \right\}.$$

Démonstration. C'est une conséquence directe de l'expression de $\eta^{\mathbf{C}}(z)$ (voir le théorème 10.3) et du théorème 2.3. ■

10.3 Conclusion

Dans ce chapitre, nous nous sommes intéressés au problème classique du calcul de racines de polynômes sous l'aspect du conditionnement et de l'erreur inverse. Étant donné un polynôme à coefficients réels, nous avons défini un conditionnement et une erreur inverse réelle, c'est-à-dire que nous autorisons seulement des perturbations réelles sur ces coefficients. Nous avons donné une formule explicite pour ces expressions.

Nous avons surtout montré qu'il y a peu de différence entre le conditionnement classique et le conditionnement réel. Par contre, il peut y avoir une différence importante entre l'erreur inverse classique et l'erreur inverse réelle.

Troisième partie

Pseudospectres et conditionnement matriciel

Chapitre 11

Pseudospectres et structuration

11.1 Introduction et notations

Le ε-pseudospectre d'une matrice A a été introduit par Trefethen [193] comme le sous-ensemble du plan complexe consistant en toutes les valeurs propres de toutes les matrices situées à une distance ε de A. Si maintenant la matrice A possède une certaine structure (par exemple Toeplitz), il semble alors naturel de n'autoriser que des perturbations ayant la même structure. Dans ce cas, on définit le ε-pseudospectre structuré d'une matrice structurée A comme le sous-ensemble du plan complexe consistant en toutes les valeurs propres de toutes les matrices structurées de même type situées à une distance ε de A.

Dans ce chapitre, nous nous intéresserons plus particulièrement aux structures vectorielles

$$\text{struct} \in \{\text{Toep}, \text{circ}, \text{Hankel}, \text{sym}\} \tag{11.1}$$

correspondant à l'ensemble des matrices Toeplitz, circulantes, Hankel et symétriques (voir la table 11.1). La table 11.2 indique le nombre de paramètres indépendants pour les structures que nous allons considérer dans ce chapitre.

Dans ce chapitre, $M_n(\mathbf{C})$ représente l'ensemble des matrices complexes de taille $n \times n$ et $M_n^{\text{struct}}(\mathbf{C})$ l'ensemble des matrices structurées complexes, avec struct dans (11.1). Nous munissons ces espaces de la norme 2 (encore appelé norme spectrale) noté $\|\cdot\|$.

Considérons maintenant une matrice $A \in M_n(\mathbf{C})$. Nous notons par $\Lambda(A)$ son spectre. Étant donné $\varepsilon > 0$, le ε-pseudospectre de la matrice $A \in M_n(\mathbf{C})$ est l'ensemble $\Lambda_\varepsilon(A)$ défini par

$$\Lambda_\varepsilon(A) = \{z \in \mathbf{C} : z \in \Lambda(X) \text{ avec } X \in M_n(\mathbf{C}) \text{ et } \|X - A\| \leqslant \varepsilon\}.$$

Étant donné une matrice $A \in M_n^{\text{struct}}(\mathbf{C})$ avec la structure struct dans (11.1), le ε-pseudospectre structuré de A est l'ensemble $\Lambda_\varepsilon^{\text{struct}}(A)$ défini par

$$\Lambda_\varepsilon^{\text{struct}}(A) = \{z \in \mathbf{C} : z \in \Lambda(X) \text{ avec } X \in M_n^{\text{struct}}(\mathbf{C}) \text{ et } \|X - A\| \leqslant \varepsilon\}.$$

Pour $A \in M_n^{\text{struct}}(\mathbf{C})$ il est clair que l'on a toujours

$$\Lambda_\varepsilon^{\text{struct}}(A) \subseteq \Lambda_\varepsilon(A).$$

Matrice Toeplitz $(t_{i-j})_{i,j=0}^{n-1}$ Matrice Hankel $(h_{i,j})_{i,j=0}^{n-1}$

$$\begin{pmatrix} t_0 & t_{-1} & \cdots & t_{1-n} \\ t_1 & t_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_{-1} \\ t_{n-1} & \cdots & t_1 & t_0 \end{pmatrix} \qquad \begin{pmatrix} h_0 & h_1 & \cdots & h_{n-1} \\ h_1 & h_2 & \iddots & h_n \\ \vdots & \iddots & \iddots & \vdots \\ h_{n-1} & h_n & \cdots & h_{2n-2} \end{pmatrix}$$

Matrice circulante $(v_i)_{i=0}^{n-1}$

$$\begin{pmatrix} v_0 & v_{n-1} & \cdots & v_1 \\ v_1 & v_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & v_{n-1} \\ v_{n-1} & \cdots & v_1 & v_0 \end{pmatrix}$$

TAB. 11.1 – Trois classes importantes de matrices structurées

Structure	générale	Toep	circ	Hankel	sym
k	n^2	$2n-1$	n	$2n-1$	$(n^2+n)/2$

TAB. 11.2 – Nombre de paramètres indépendants

Nous sommes intéressé par trouver les structures pour lesquelles on a l'égalité entre les pseudospectres structurés et non structurés. Pour les structures classiques où il n'y a pas égalité, nous dégageons les différentes relations qui peuvent exister entre ces deux pseudospectres.

À notre connaissance, le pseudospectre structuré (appelé en anglais « spectral value sets ») a été défini et étudié pour la première fois avec des perturbations de la forme

$$A \rightsquigarrow A + \Delta A = A + D\Theta E, \quad \Theta \in M_{l,q}(\mathbf{C}),$$

où $D \in M_{n,l}(\mathbf{C})$, $E \in M_{q,n}(\mathbf{C})$ sont des matrices fixées définissant la structure de la perturbation (voir [89, 191, 18]). La définition de pseudospectre structuré que nous utilisons dans ce chapitre est du à Böttcher, Grudsky et Kozak [20] pour la structure Toeplitz. Ils l'appelèrent le ε-pseudospectre « Toeplitz » dans [20] et le pseudospectre Toeplitz-structuré dans [19]. Dans [20], ils considèrent des matrices Toeplitz par bandes et donc se restreignent à définir $\Lambda_\varepsilon^{\mathrm{Toep}[r,s]}(A)$ pour $A \in M_n^{\mathrm{Toep}[r,s]}(\mathbf{C})$ où $\mathrm{Toep}[r,s]$ est la matrice Toeplitz avec au plus r superdiagonales non nulles et au plus s sousdiagonales non nulles. Ils montrent que $\Lambda_\varepsilon(A)$ peut être différent de $\Lambda_\varepsilon^{\mathrm{Toep}[r,s]}(A)$. Dans ce chapitre, nous montrons l'égalité quand $r = s = n$. De plus, nous étendons la définition à d'autres structures, telle que les structures Hankel et symétriques.

Le reste du chapitre est organisé de la manière suivante. Dans la section 11.2, nous rappelons des résultats sur la distance structurée à la singularité. Dans la section 11.3,

nous montrons que pour struct $\in \{\text{Toep}, \text{circ}, \text{Hankel}, \text{sym}\}$, le pseudospectre structuré est égal au pseudospectre non structuré. Nous étudions ensuite le cas des structures Hermitiennes et anti-Hermitiennes. Nous montrons qu'il n'y a pas égalité entre le pseudospectre structuré et non structuré pour ces structures. Dans la section 11.4, nous généralisons les résultats précédents au cas du pseudospectre de matrices polynomiales avec struct $\in \{\text{Toep}, \text{circ}, \text{Hankel}, \text{sym}\}$. Enfin nous donnons une formule calculable pour le pseudospectre structuré de matrices polynomiales réelles.

11.2 Distance structurée à la singularité

Dans cette section, nous rappelons quelques résultats concernant le distance structurée à la singularité. Étant donnée une matrice non singulière $A \in M_n(\mathbf{C})$, on définit la distance à la singularité par

$$d(A) = \min\{\|\Delta A\| : A + \Delta A \text{ singulière}, \Delta A \in M_n(\mathbf{C})\}. \tag{11.2}$$

Pour une matrice non singulière $A \in M_n^{\text{struct}}(\mathbf{C})$, on définit la distance structurée à la singularité par

$$d^{\text{struct}}(A) = \min\{\|\Delta A\| : A + \Delta A \text{ singulière}, \Delta A \in M_n^{\text{struct}}(\mathbf{C})\}. \tag{11.3}$$

Rump a prouvé dans [172, Thm 12.2] que les deux distances $d(A)$ et $d^{\text{struct}}(A)$ étaient égales pour struct $\in \{\text{Toep}, \text{circ}, \text{Hankel}\}$.

Théorème 11.1 (Rump [172, Thm 12.2]). *Étant donnée une matrice non singulière $A \in M_n^{\text{struct}}(\mathbf{C})$ avec* struct $\in \{\text{Toep}, \text{circ}, \text{Hankel}\}$. *Alors on a*

$$d(A) = d^{\text{struct}}(A) = \|A^{-1}\|^{-1} = \sigma_{\min}(A).$$

Le nombre $\sigma_{\min}(A)$ représente la plus petite valeur singulière de A. On a la même propriété avec la structure symétrique. Avant d'énoncer le résultat, nous avons besoin des lemmes suivant.

Lemme 11.1 (factorisation de Takagi). *Soit A une matrice symétrique complexe ($A^T = A$), alors il existe une matrice unitaire U et une matrice diagonale à éléments réels positifs $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$ telles que $A = U\Sigma U^T$.*

On trouvera une preuve de ce résultat dans [99, Cor. 4.4.4].

Lemme 11.2 (Rump [172, Lem. 10.1]). *Soit $x \in \mathbf{C}^n$ un vecteur donné. Alors il existe une matrice symétrique complexe A telle que $Ax = \overline{x}$ et $\|A\| = 1$.*

Le résultat suivant est issu de Tisseur et Graillat [189].

Théorème 11.2 (Tisseur et Graillat [189]). *Soit A une matrice non singulière de $M_n^{\text{struct}}(\mathbf{C})$ avec* struct $=$ sym. *Alors*

$$d(A) = d^{\text{struct}}(A) = \|A^{-1}\|^{-1} = \sigma_{\min}(A).$$

Démonstration. On a évidemment $d^{\text{struct}}(A) \geqslant d(A) = \|A^{-1}\|^{-1} = \sigma_{\min}(A)$, et ainsi il nous reste seulement à montrer que $(A+\Delta A)x = 0$ pour un $x \neq 0$ et ΔA symétrique avec $\|\Delta A\| = \sigma_{\min}(A)$. Soit $A = U\Sigma U^T$ la factorisation de Takagi de A où U est unitaire et Σ est diagonale avec des éléments strictement positives (voir le lemme 11.1). Soit x la colonne de U correspondant au plus petit élément diagonal de Σ. Alors $A\overline{x} = \sigma_{\min}(A)x$. D'après le lemme 11.2 il existe une matrice symétrique complexe C telle que $C\overline{x} = x$ et $\|C\| = 1$. Posons $\Delta A = -\sigma_{\min}(A)C$. Il vient alors que ΔA est symétrique, $\|\Delta A\| = \sigma_{\min}(A)$ et

$$(A + \Delta A)\overline{x} = \sigma_{\min}(A)x - \sigma_{\min}(A)x = 0$$

si bien que $A + \Delta A$ est singulière. ■

11.3 Le pseudospectre structuré égale le pseudospectre non structuré

Le lemme suivant établit un lien entre le ε-pseudospectre et la distance à la singularité (voir [195]).

Lemme 11.3. *Étant donnés $\varepsilon > 0$ et $A \in M_n(\mathbf{C})$, le ε-pseudospectre vérifie*

$$\Lambda_\varepsilon(A) = \{z \in \mathbf{C} : d(A - zI) \leqslant \varepsilon\}.$$

Dans cette section, nous considérons les structures suivantes

$$\text{struct} \in \{\text{Toep}, \text{circ}, \text{sym}\}. \tag{11.4}$$

Comme nous l'avons vu précédemment, nous avons $d(A) = d^{\text{struct}}(A)$ pour $A \in M_n^{\text{struct}}(\mathbf{C})$. Par conséquent, il nous suffit de prouver que

$$\Lambda_\varepsilon^{\text{struct}}(A) = \{z \in \mathbf{C} : d^{\text{struct}}(A - zI) \leqslant \varepsilon\}$$

pour conclure que $\Lambda_\varepsilon(A) = \Lambda_\varepsilon^{\text{struct}}(A)$ pour une matrice $A \in M_n^{\text{struct}}(\mathbf{C})$. C'est le but du lemme suivant.

Lemme 11.4. *Étant donnés $\varepsilon > 0$ et $A \in M_n^{\text{struct}}(\mathbf{C})$ avec* struct *dans (11.4), le ε-pseudospectre structuré satisfait*

$$\Lambda_\varepsilon^{\text{struct}}(A) = \{z \in \mathbf{C} : d^{\text{struct}}(A - zI) \leqslant \varepsilon\}.$$

La preuve est très similaire à celle du lemme 11.3 mais nous devons faire attention à conserver la structure.

Démonstration. Soit $z \in \Lambda_\varepsilon^{\text{struct}}(A)$. Cela signifie qu'il existe $X \in M_n^{\text{struct}}(\mathbf{C})$ tel que $z \in \Lambda(X)$ et $\|X - A\| \leqslant \varepsilon$. Par conséquent la matrice $X - zI$ est singulière et

$$\|(X - zI) - (A - zI)\| = \|X - A\| \leqslant \varepsilon.$$

De plus, puisque pour les structures considérées $zI \in M_n^{\text{struct}}(\mathbf{C})$ pour $z \in \mathbf{C}$, nous avons $X - zI \in M_n^{\text{struct}}(\mathbf{C})$.Par définition de la distance à la singularité, on a

$$d^{\text{struct}}(A - zI) \leqslant \varepsilon.$$

Réciproquement, soit $z \in C$ tel que $d^{\text{struct}}(A - zI) \leqslant \varepsilon$. Alors il existe $X \in M_n^{\text{struct}}(\mathbf{C})$ tel que X soit singulière et $\|(A - zI) - X\| \leqslant \varepsilon$. Posons $Y = X + zI$. On remarque alors que z est une valeur propre de Y puisque $Y - zI = X$ est singulière. De plus, nous avons

$$\|Y - A\| = \|(A - zI) - X\| \leqslant \varepsilon$$

et $Y \in M_n^{\text{struct}}(\mathbf{C})$ puisque $zI \in M_n^{\text{struct}}(\mathbf{C})$ pour $z \in \mathbf{C}$. On en déduit que $z \in \Lambda(Y)$ avec $\|Y - A\| \leqslant \varepsilon$, c'est-à-dire que $z \in \Lambda_\varepsilon^{\text{struct}}(A)$. ■

D'après les lemmes 11.3 et 11.4 et les théorèmes 11.1 et 11.2 on déduit le théorème suivant.

Théorème 11.3. *Étant donnés $\varepsilon > 0$ et $A \in M_n^{\text{struct}}(\mathbf{C})$ avec* struct $\in \{\text{Toep}, \text{circ}, \text{sym}\}$*, le ε-pseudospectre et le ε-pseudospectre structuré sont égaux, c'est-à-dire,*

$$\Lambda_\varepsilon^{\text{struct}}(A) = \Lambda_\varepsilon(A).$$

Le théorème 11.1 est encore vrai pour les structures Hermitiennes et anti-Hermitienne Cependant, la preuve du lemme 11.4 donnée ci-dessus ne s'adapte pas à ces deux structures (ni à celle Hankel) puisque les matrices scalaires (zI pour $z \in \mathbf{C}$) ne sont ni Hermitiennes ni anti-Hermitiennes en général.

En fait, nous n'avons pas égalité entre le pseudospectre structuré et le pseudospectre non structuré pour ces structures. En effet, les matrices Hermitienne et anti-Hermitienne sont normales, et si $A \in M_n(\mathbf{C})$ est normale alors $\Lambda_\varepsilon(A) = \{z \in \mathbf{C} : \text{dist}(z, \Lambda(A)) \leqslant \varepsilon\}$ (voir [193]). Par conséquent, $\Lambda_\varepsilon(A)$ contient un sous-ensemble ouvert de $\mathbf{C}$. Mais si A est Hermitienne alors évidemment $\Lambda_\varepsilon^{\text{herm}}(A) \subset \mathbf{R}$, tandis que si A est anti-Hermitienne il est facile de voir que $\Lambda_\varepsilon^{\text{skewherm}}(A) \subset i\mathbf{R}$. Ceci montre que pour les matrices Hermitiennes et anti-Hermitiennes le pseudospectre est toujours strictement plus grand que le pseudospectre structuré. Il est clair que $\Lambda_\varepsilon^{\text{herm}}(A) \subset \Lambda_\varepsilon(A) \cap \mathbf{R}$. Soit $z \in \Lambda_\varepsilon(A) \cap \mathbf{R}$. Puisque maintenant zI est Hermitienne, on en déduit que $d^{\text{herm}}(A - zI) = d(A - zI)$ et donc $z \in \Lambda_\varepsilon^{\text{herm}}(A)$. Par conséquent, $\Lambda_\varepsilon^{\text{herm}}(A) = \Lambda_\varepsilon(A) \cap \mathbf{R}$. En reprenant les mêmes arguments, on conclut aussi que $\Lambda_\varepsilon^{\text{skewherm}}(A) = \Lambda_\varepsilon(A) \cap i\mathbf{R}$.

En fait, l'égalité $\Lambda_\varepsilon^{\text{Hankel}}(A) = \Lambda_\varepsilon(A)$ reste valable pour A dans $M_n^{\text{Hankel}}(\mathbf{C})$. Pour voir cela, soit $z \in \Lambda_\varepsilon(A)$. Comme dans la preuve du théorème 11.2, considérons la factorisation de Takagi $A - zI = U\Sigma U^T$ et prenons $x \neq 0$ tel que $(A - zI)\overline{x} = \sigma_{\min}(A - zI)x$. Rump [172] a montré que la matrice A du lemme 11.2 pouvait être choisi de type Hankel. Ainsi, il existe une matrice $C \in M_n^{\text{Hankel}}(\mathbf{C})$ telle que $C\overline{x} = x$ et $\|C\| = 1$. On en déduit alors que $\Delta A := -\sigma_{\min}(A - zI)C$ est une matrice de Hankel et que $(A - zI + \Delta A)\overline{x} = 0$. En conclusion, on a $z \in \Lambda_\varepsilon^{\text{Hankel}}(A)$.

11.4 Pseudospectre structuré de matrices polynomiales

Dans cette section, nous nous intéressons au pseudospectre de matrices polynomiales. Nous prouvons un résultat analogue au théorème 11.3 pour les matrices polynomiales dans la première sous-section. Le seconde sous-section concerne le pseudospectre de matrices polynomiales réelles en ne prenant en compte que les perturbations réelles.

11.4.1 Pseudospectre structuré de matrices polynomiales complexes

Le problème du calcul des valeurs propres d'une matrice polynomiale est de trouver la solution $(x, \lambda) \in \mathbf{C}^n \times \mathbf{C}$ de

$$P(\lambda)x = 0,$$

où

$$P(\lambda) = \lambda^m A_m + \lambda^{m-1} A_{m-1} + \cdots + A_0,$$

avec $A_k \in M_n(\mathbf{C})$, $k = 0 : m$. Si $x \neq 0$ alors λ est appelée une valeur propre de P et x un vecteur propre associé à la valeur propre λ. L'ensemble des valeurs propres de P est noté $\Lambda(P)$. Quand A_m est non singulière, P est dit *régulier* et a mn valeurs propres. Dans la suite, nous supposons toujours que P est régulier. Définissons

$$\Delta P(\lambda) = \lambda^m \Delta A_m + \lambda^{m-1} \Delta A_{m-1} + \cdots + \Delta A_0,$$

où $\Delta A_k \in M_n(\mathbf{C})$. On définit le ε-pseudospectre de P par

$$\Lambda_\varepsilon(P) = \{\lambda \in \mathbf{C} : (P(\lambda) + \Delta P(\lambda))x = 0 \text{ pour un } x \neq 0 \\ \text{avec } \|\Delta A_k\| \leqslant \alpha_k \varepsilon, k = 0 : m\}.$$

Les paramètres positifs $\alpha_1, \ldots, \alpha_m$ permettent une pondération sur les perturbations de chaque matrices. En la définition précédente nous supposons aussi que toutes les matrices polynomiales $P(\lambda) + \Delta P(\lambda)$ sont aussi régulières. Le lemme suivant est une reformulation du Lemme 2.1 de [191].

Lemme 11.5. *Nous avons*

$$\Lambda_\varepsilon(P) = \{\lambda \in \mathbf{C} : d(P(\lambda)) \leqslant \varepsilon p(|\lambda|)\},$$

où $p(x) = \sum_{k=0}^m \alpha_k x^k$.

Démonstration. Soit λ un élément de $\Lambda_\varepsilon(P)$. Il existe alors $\Delta P(\lambda) \in M_n(\mathbf{C})$ tel que $\|\Delta A_k\| \leqslant \alpha_k \varepsilon, k = 0 : m$ et $P(\lambda) + \Delta P(\lambda)$ est singulière. On déduit alors de la définition de la distance d que $d(P(\lambda)) \leqslant \|\Delta P(\lambda)\|$. Puisque

$$\|\Delta P(\lambda)\| \leqslant \sum_{k=0}^m |\lambda|^k \alpha_k \varepsilon = \varepsilon p(|\lambda|),$$

on a $d(P(\lambda)) \leqslant \varepsilon p(|\lambda|)$.

Réciproquement, soit $\lambda \in \mathbf{C}$ tel que $d(P(\lambda)) \leqslant \varepsilon p(|\lambda|)$. Cela signifie qu'il existe $X \in M_n(\mathbf{C})$ tel que $\|X\| \leqslant \varepsilon p(|\lambda|)$ et $P(\lambda) + X$ est singulière. Définissons ΔA_k par

$$\Delta A_k = \operatorname{sign}(\lambda^k)\alpha_k p(|\lambda|)^{-1} X,$$

où pour un nombre complexe z on a définit

$$\operatorname{sign}(z) = \begin{cases} \overline{z}/|z|, & z \neq 0, \\ 0, & z = 0. \end{cases}$$

On a alors

$$\Delta P(\lambda) = \sum_{k=0}^{m} \lambda^k \Delta A_k = \left(\sum_{k=0}^{m} |\lambda|^k \alpha_k p(|\lambda|)^{-1} X \right) = X,$$

et $\|\Delta A_k\| \leqslant \alpha_k \varepsilon$, $k = 0 : m$. Ainsi, on a $\lambda \in \Lambda_\varepsilon(P)$. ■

Nous supposons maintenant que les matrices A_k possèdent toutes la même structure appartenant à

$$\text{struct} \in \{\text{Toep}, \text{circ}, \text{Hankel}, \text{sym}\}. \tag{11.5}$$

Nous supposons aussi que toutes les matrices ΔA_k, $k = 0 : n$, ont la même structure que les matrices A_k. Soit

$$P(\lambda) = \lambda^m A_m + \lambda^{m-1} A_{m-1} + \cdots + A_0,$$

avec $A_k \in M_n^{\text{struct}}(\mathbf{C})$, $k = 0 : m$ et

$$\Delta P(\lambda) = \lambda^m \Delta A_m + \lambda^{m-1} \Delta A_{m-1} + \cdots + \Delta A_0,$$

avec $\Delta A_k \in M_n^{\text{struct}}(\mathbf{C})$. On remarque aisément que $P(\lambda)$ et $\Delta P(\lambda)$ appartiennent à $M_n^{\text{struct}}(\mathbf{C})$. On définit le ε-pseudospectre structuré de P par

$$\begin{aligned} \Lambda_\varepsilon^{\text{struct}}(P) = \{\lambda \in \mathbf{C} : (P(\lambda) + \Delta P(\lambda))x = 0 \text{ pour un } x \neq 0 \\ \text{avec } \Delta A_k \in M_n^{\text{struct}}(\mathbf{C}), \|\Delta A_k\| \leqslant \alpha_k \varepsilon, k = 0 : n\}. \end{aligned}$$

Le lemme suivant est la version structurée du lemme 11.5.

Lemme 11.6. *Pour* struct *dans (11.5) on a*

$$\Lambda_\varepsilon^{\text{struct}}(P) = \{\lambda \in \mathbf{C} : d^{\text{struct}}(P(\lambda)) \leqslant \varepsilon p(|\lambda|)\},$$

où $p(x) = \sum_{k=0}^{n} \alpha_k x^k$.

Démonstration. Soit λ un élément $\Lambda_\varepsilon^{\text{struct}}(P)$. Il existe donc $\Delta P(\lambda) \in M_n^{\text{struct}}(\mathbf{C})$ tel que $\Delta A_k \in M_n^{\text{struct}}(\mathbf{C})$, $\|\Delta A_k\| \leqslant \alpha_k \varepsilon, k = 0 : m$ et $P(\lambda) + \Delta P(\lambda)$ est singulière. De la définition de la distance d^{struct}, on déduit que $d^{\text{struct}}(P(\lambda)) \leqslant \|\Delta P(\lambda)\|$. Puisque

$$\|\Delta P(\lambda)\| \leqslant \sum_{k=0}^{m} |\lambda|^k \alpha_k \varepsilon = \varepsilon p(|\lambda|),$$

on a $d^{\text{struct}}(P(\lambda)) \leqslant \varepsilon p(|\lambda|)$.

Réciproquement, soit $\lambda \in \mathbf{C}$ tel que $d^{\text{struct}}(P(\lambda)) \leqslant \varepsilon p(|\lambda|)$. Cela signifie qu'il existe $X \in M_n^{\text{struct}}(\mathbf{C})$ tel que $\|X\| \leqslant \varepsilon p(|\lambda|)$ et $P(\lambda) + X$ est singulière. Définissons ΔA_k par

$$\Delta A_k = \text{sign}(\lambda^k)\alpha_k p(|\lambda|)^{-1} X.$$

On a lors, $\Delta A_k \in M_n^{\text{struct}}(\mathbf{C})$,

$$\Delta P(\lambda) = \sum_{k=0}^{m} \lambda^k \Delta A_k = \left(\sum_{k=0}^{m} |\lambda|^k \alpha_k p(|\lambda|)^{-1} X \right) = X,$$

et $\|\Delta A_k\| \leqslant \alpha_k \varepsilon$, $k = 0 : m$. Par conséquent, $\lambda \in \Lambda_\varepsilon(P)$. ■

Des lemmes 11.5 et 11.6 et des théorèmes 11.1 et 11.2 on déduit le théorème suivant pour struct dans (11.5).

Théorème 11.4. *Si $\varepsilon > 0$ et $P(\lambda) = \lambda^m A_m + \lambda^{m-1} A_{m-1} + \cdots + A_0$ est une matrice polynomiale avec $A_k \in M_n^{\text{struct}}(\mathbf{C})$, $k = 0 : m$ et*

$$\text{struct} \in \{\text{Toep}, \text{circ}, \text{Hankel}, \text{sym}\},$$

alors le ε-pseudospectre et le ε-pseudospectre structuré sont égaux, c'est-à-dire,

$$\Lambda_\varepsilon^{\text{struct}}(P) = \Lambda_\varepsilon(P).$$

11.4.2 Pseudospectre structuré de matrices polynomiales réelles

Dans cette sous-section, nous considérons

$$P(\lambda) = \lambda^m A_m + \lambda^{m-1} A_{m-1} + \cdots + A_0,$$

avec $A_k \in M_n(\mathbf{R})$, $k = 0 : m$ et

$$\Delta P(\lambda) = \lambda^m \Delta A_m + \lambda^{m-1} \Delta A_{m-1} + \cdots + \Delta A_0,$$

avec $\Delta A_k \in M_n(\mathbf{R})$. Nous supposons que $P(\lambda)$ est sujette à des perturbations structurées qui s'expriment sous la forme

$$[\Delta A_0, \ldots, \Delta A_m] = D\Theta[E_0, \ldots, E_m],$$

avec $D \in M_{n,1}(\mathbf{R})$, $\Theta \in M_{1,t}(\mathbf{R})$ et $E_k \in M_{t,n}(\mathbf{R})$, $k = 0 : m$. Ce type de perturbations apparaît naturellement en théorie du contrôle. Par commodité, nous introduisons les notations suivantes

$$E(\lambda) = E[I_n, \lambda I_n, \ldots, \lambda^m I_n]^T = \lambda^m E_m + \lambda^{m-1} E_{m-1} + \cdots + E_0,$$

et

$$G(\lambda) = E(\lambda)P(\lambda)^{-1}D = G_R(\lambda) + iG_I(\lambda), \quad G_R(\lambda), G_I(\lambda) \in \mathbf{R}^t.$$

On définit le ε-pseudospectre structuré par

$$\Lambda_\varepsilon(P) = \{\lambda \in \mathbf{C} : (P(\lambda) + D\Theta E(\lambda))x = 0 \text{ pour un } x \neq 0, \|\Theta\| \leqslant \varepsilon\}.$$

Nous supposons de plus que les matrices polynomiale P et $P(\lambda)+D\Theta E(\lambda)$ sont régulières. Pour $x, y \in \mathbf{R}^t$, nous notons

$$\operatorname{dist}(x, \mathbf{R}y) = \inf_{\alpha\in\mathbf{R}} \|x - \alpha y\|,$$

la distance du point x à la droite $\mathbf{R}y = \{\alpha y, \alpha \in \mathbf{R}\}$. Le théorème suivant fournit une caractérisation calculable du pseudospectre structuré.

Théorème 11.5. *Nous avons*

$$\Lambda_\varepsilon(P) = \{\lambda \in \mathbf{C} \setminus \Lambda(P) : \operatorname{dist}(G_R(\lambda), \mathbf{R}G_I(\lambda)) \geqslant 1/\varepsilon\} \cup \Lambda(P).$$

Démonstration. S'il existe $x \neq 0$ tel que $(P(\lambda) + D\Theta E(\lambda))x = 0$ alors

$$x = -P(\lambda)^{-1}D\Theta E(\lambda)x \quad \text{et} \quad \Theta E(\lambda)x = -\Theta E(\lambda)P(\lambda)^{-1}D\Theta E(\lambda)x.$$

Notons $u = \Theta E(\lambda)x \in \mathbf{C}$, $u = u_1 + iu_2$, $(u_1, u_2) \in \mathbf{R}^2$. Il est clair que $u \neq 0$ puisque $\lambda \notin \Lambda(P)$. En utilisant ces notations, on obtient

$$u = -\Theta G(\lambda)u,$$

que l'on peut réécrire en termes réels,

$$\begin{cases} u_1 = -\Theta G_R(\lambda)u_1 + \Theta G_I(\lambda)u_2, \\ u_2 = -\Theta G_R(\lambda)u_2 - \Theta G_I(\lambda)u_1. \end{cases}$$

Ces équations sont équivalentes à

$$\begin{cases} (1 + \Theta G_R(\lambda))u_1 - \Theta G_I(\lambda)u_2 = 0, \\ -\Theta G_I(\lambda)u_1 - (1 + \Theta G_R(\lambda))u_2 = 0. \end{cases}$$

Puisque $(u_1, u_2) \neq (0, 0)$, le système a une solution non triviale. On en déduit que le déterminant du système s'annule. Un calcul simple montre que ce déterminant est égal à $(1 + \Theta G_R(\lambda))^2 + (\Theta G_I(\lambda))^2$. On en conclut que Θ satisfait les équations ci-dessus si et seulement si

$$\Theta G_I(\lambda) = 0 \quad \text{et} \quad \Theta G_R(\lambda) = -1.$$

En conséquence, $\Theta(G_R(\lambda) - \alpha G_I(\lambda)) = -1$ pour tout $\alpha \in \mathbf{R}$, si bien que nous avons $1 \leqslant \varepsilon\|G_R(\lambda) - \alpha G_I(\lambda)\|$. Ainsi, on a bien

$$\operatorname{dist}(G_R(\lambda), \mathbf{R}G_I(\lambda)) \geqslant 1/\varepsilon.$$

Réciproquement, supposons que $\operatorname{dist}(G_R(\lambda), \mathbf{R}G_I(\lambda)) \geqslant 1/\varepsilon$. Par un théorème de dualité (voir [93]) il existe un vecteur $z \in \mathbf{R}^t$, $\|z\| = 1$ tel que

$$\begin{cases} z^T G_R(\lambda) = \operatorname{dist}(G_R(\lambda), \mathbf{R}G_I(\lambda)), \\ z^T G_I(\lambda) = 0. \end{cases}$$

Posons $\Theta = -\operatorname{dist}(G_R(\lambda), \mathbf{R}G_I(\lambda))^{-1}z$ et $x = P(\lambda)^{-1}D$. Avec ces notations, nous avons $(P(\lambda) + D\Theta E(\lambda))x = 0$. ∎

11.5 Conclusion

Dans ce chapitre, nous avons montré que le pseudospectre structuré était égal au pseudospectre non structuré pour les structures suivante : Toeplitz, circulante, Hankel et symétrique. Nous avons montré que ce résultat ne pouvait s'étendre aux structures Hermitiennes et anti-Hermitienne. Nous avons généralisé ces résultats au pseudospectre de matrices polynomiales pour les mêmes structures. Nous avons de plus donné une formule pour le calcul du pseudospectre structuré de matrices polynomiales réelles.

Chapitre 12

Conditionnement structuré de problèmes matriciels

12.1 Motivations

Il y a actuellement un intérêt croissant dans l'analyse des perturbations structurées (voir par exemple [27], [83], [172] et leurs références) en grande partie dû à un développement substantiel d'algorithmes pour des problèmes structurés. Pour ces problème, il est souvent plus approprié de définir un conditionnement qui mesure la sensibilité aux perturbations structurées. Les perturbations structurées sont en général plus difficiles à manipuler que les perturbations non structurées et l'analyse de l'erreur inverse d'un algorithme est aussi plus compliquée quand elle concerne des données structurées. Notre but est ici d'identifier des classes de matrices pour lesquelles le rapport entre le conditionnement structuré et non-structuré ainsi que l'erreur inverse structurée et non-structurée est proche de un. Par exemple, on sait que pour les systèmes linéaires symétriques ainsi que pour la distance à la singularité, il n'y a pas de différences si les perturbations sont restreintes à être symétriques [82]. Rump [172] a montré récemment qu'il y a les même types de comportement avec les structures anti-symétriques et persymétriques. Par conséquent, dans ces cas, l'analyse usuelle avec des perturbations non-structurées est suffisante. Dans ce chapitre, nous unifions et étendons ces résultats à d'autres structures.

Les matrices structurées que nous considérons ici appartiennent aux algèbres de Lie et de Jordan [5] associées à un produit scalaire orthosymétrique et unitaire. Cela inclut par exemple les matrices symétriques, symétriques complexes, anti-symétriques complexes, pseudo-symétriques, pseudo-anti-symétriques, per-symétriques, per-anti-symétriques, hamiltoniennes, anti-hamiltoniennes, hermitiennes, anti-hermitiennes et J-hermitiennes.

Le chapitre est organisé de la façon suivante. Dans la section 12.2, nous commençons par mettre en place les notations et nous rappelons les outils de base nécessaires. Dans la section 12.3, nous étudions le conditionnement pour l'inverse d'une matrice et pour les systèmes linéaires ainsi que la distance à la singularité. Nous montrons que pour les matrices appartenant à une algèbre de Lie ou de Jordan d'un produit scalaire ortho-

symétrique et unitaire, les conditionnements structurés et non-structurés sont égaux ou dans un petit facteur l'un de l'autre. Nous montrons aussi que la distance à la singularité ne change pas si nous forçons les perturbations à être structurées. Ainsi pour ces classes de matrices, l'analyse de perturbation usuelle (c'est-à-dire non-structurée) est suffisante. Dans la section 12.4, nous considérons l'erreur inverse structurée à la fois pour les systèmes linéaires et pour les problèmes de valeurs propres. Nous donnons des formules explicites pour l'erreur inverse structurée quand elles existent et nous donnons un ratio entre l'erreur inverse structurée et non-structurée.

On note par $\mathbf{K}$ le corps $\mathbf{R}$ ou $\mathbf{C}$, par $\|\cdot\|$ la norme 2 définie par

$$\|A\|_2 = \rho(A^*A)^{1/2} = \sigma_{\max}(A)$$

et par $\|\cdot\|_F$ la norme de Frobenius

$$\|A\|_F^2 = \sum_{i=1}^{n} \|Av_i\|_2^2 = \mathrm{trace}(A^*A), \tag{12.1}$$

avec $A \in \mathbf{K}^{n\times n}$, $\sigma_{\max}$ la plus grande valeur singulière A, ρ le rayon spectral et $\{v_1, \ldots, v_n\}$ une base orthonormé de $\mathbf{K}^n$.

On remarque que ces deux normes sont unitairement invariantes, c.-à-d., $\|UAV\| = \|A\|$ pour U et V des matrices unitaires. Nous utiliserons aussi fréquemment le fait $\|A^{-1}\|_2^{-1} = \sigma_{\min}(A)$.

12.2 Préliminaires

Nous commençons par donner les définitions et les propriétés de base des produits scalaires et des classes de matrices structurées qui leur sont associées. Nous donnons aussi un certain nombre de résultats auxiliaires sur ces matrices structurées.

12.2.1 Matrices structurées dans les espaces quadratiques

Le terme *produit scalaire* sera utilisé pour se référer à toute forme bilinéaire ou sesquilinéaire non dégénérée $\langle\cdot,\cdot\rangle$ sur $\mathbf{K}^n$. Toute forme bilinéaire réelle ou complexe $\langle\cdot,\cdot\rangle$ a une unique représentation matricielle donnée par $\langle\cdot,\cdot\rangle = x^TMy$, tandis qu'une forme sesquilinéaire peut être représentée par $\langle\cdot,\cdot\rangle = x^*My$, où la matrice M est inversible. Nous noterons $\langle\cdot,\cdot\rangle$ par $\langle\cdot,\cdot\rangle_M$ quand cela sera nécessaire.

Une forme bilinéaire est symétrique si $\langle x, y\rangle = \langle y, x\rangle$ et anti-symétrique si $\langle x, y\rangle = -\langle y, x\rangle$. Ainsi pour une forme symétrique on a $M = M^T$ et pour une forme anti-symétrique $M = -M^T$. Une forme sesquilinéaire est hermitienne si $\langle x, y\rangle = \overline{\langle y, x\rangle}$ et anti-hermitienne si $\langle x, y\rangle = -\overline{\langle y, x\rangle}$. Les matrices associées à de telles formes sont respectivement hermitiennes et anti-hermitiennes.

À chaque produit scalaire correspond une notion d'*adjoint*, généralisant l'idée de la transposition ^{T} ou de le transconjugaison *. Cela signifie que pour toute matrice $A \in \mathbf{K}^{n\times n}$

TAB. 12.1 – Échantillons de matrices structurées associées à un produit scalaire $\langle \cdot, \cdot \rangle_{\mathbf{M}}$, où M est la matrice définissant le produit scalaire.

Espace	M	**Groupe d'automorphismes** $\mathbb{G} = \{G : G^\star = G^{-1}\}$	**Algèbre de Jordan** $\mathbb{J} = \{S : S^\star = S\}$	**Algèbre de Lie** $\mathbb{L} = \{K : K^\star = -K\}$
Formes bilinéaires				
$\mathbf{R}^n$	I	Orthogonales réelles	Symétriques	Anti-symétriques
$\mathbf{C}^n$	I	Orthogonales complexes	Symétriques complexes	Anti-symétriques complexes
$\mathbf{R}^n$	$\Sigma_{p,q}$	Pseudo-orthogonales[a]	Pseudo symétriques	Pseudo anti-symétriques
$\mathbf{C}^n$	$\Sigma_{p,q}$	pseudo-orthogonales complexes	pseudo-symétriques complexes	Pseudo-anti-symétriques complexes
$\mathbf{R}^n$	R	Perplectiques réelles	Persymétriques	Per-anti-symétriques
$\mathbf{R}^{2n}$	J	Symplectiques réelles	Anti-hamiltoniennes	Hamiltonienne
$\mathbf{C}^{2n}$	J	Symplectiques complexes	J-anti-symétriques complexes	J-symétriques complexes
Formes sesquilinéaires				
$\mathbf{C}^n$	I	Unitaires	Hermitiennes	Anti-hermitiennes
$\mathbf{C}^n$	$\Sigma_{p,q}$	Pseudo-unitaires[b]	Pseudo-hermitiennes	Pseudo-anti-hermitiennes
$\mathbf{C}^{2n}$	J	Symplectiques conjuguées	J-anti-hermitiennes	J-hermitiennes

Ici, $R = \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix}$ et $\Sigma_{p,q} = \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} \in \mathbf{R}^{n \times n}$ sont symériques et $J = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}$ est anti-symétrique.

[a] Les physiciens utilisent le groupe pseudo-orthogonal avec $\Sigma_{3,1} = \begin{bmatrix} I_3 & \\ & -1 \end{bmatrix}$ et l'appelle le groupe de Lorentz.

[b] Les matrices pseudo-unitaires sont quelques fois appelées $\Sigma_{p,q}$-unitaires, ou hypernormales en traitement du signal.

il existe une unique matrice adjointe $A^\star$ par rapport au produit scalaire définie par $\langle Ax, y\rangle_{\text{M}} = \langle x, A^\star y\rangle_{\text{M}}$ pour tous x et y dans $\mathbf{K}^n$. En terme matriciel la matrice adjointe est donnée par

$$A^\star = \begin{cases} M^{-1}A^T M & \text{pour les formes bilinéaires,} \\ M^{-1}A^* M & \text{pour les formes sesquilinéaires.} \end{cases}$$

On associe à un produit scalaire $\langle\cdot,\cdot\rangle_{\text{M}}$ trois ensembles particuliers : l'algèbre de Lie $\mathbb{L}$, l'algèbre de Jordan $\mathbb{J}$ et le groupe d'automorphismes $\mathbb{G}$ définis respectivement par

$$\begin{aligned} \mathbb{L} = &\quad \{L \in \mathbf{K}^{n\times n} : \langle Lx, y\rangle_{\text{M}} = -\langle x, Ly\rangle_{\text{M}}\} &&= \left\{L \in \mathbf{K}^{n\times n} : L^\star = -L\right\}, \\ \mathbb{J} = &\quad \{S \in \mathbf{K}^{n\times n} : \langle Sx, y\rangle_{\text{M}} = \langle x, Sy\rangle_{\text{M}}\} &&= \left\{S \in \mathbf{K}^{n\times n} : S^\star = S\right\}, \\ \mathbb{G} = &\quad \{G \in \mathbf{K}^{n\times n} : \langle Gx, Gy\rangle_{\text{M}} = \langle x, y\rangle_{\text{M}}\} &&= \{G \in \mathbf{K}^{n\times n} :\ G^\star = G^{-1}\}. \end{aligned}$$

Les ensembles $\mathbb{L}$ et $\mathbb{J}$ sont des espaces vectoriels. Ils ne sont pas fermés pour la multiplication mais le sont pour l'inversion : si $A \in \mathbb{S}$ est inversible avec $\mathbb{S} = \mathbb{L}$ ou $\mathbb{S} = \mathbb{J}$, alors $A^{-1} \in \mathbb{S}$. L'ensemble $\mathbb{G}$ muni de la multiplication forme un groupe de Lie.

Dans le reste du chapitre, nous nous concentrons sur les matrices structurées qui appartiennent soit à une algèbre de Lie, soit à une algèbre de Jordan, soit à un groupe d'automorphismes dont le produit scalaire est *orthosymétrique*, c'est-à-dire,

$$M = \begin{cases} \pm M^T, & \text{(formes bilinéaires),} \\ \beta M^*, \ |\beta| = 1, & \text{(formes sesquilinéaires).} \end{cases}$$

(Voir [135, Définition A.4] pour une liste de propriétés équivalentes.) Cela inclut par exemple les matrices structurées listées dans la table 12.1, toutes étant issues d'un produit scalaire orthosymétrique avec $\beta = 1$. On remarque que les formes bilinéaires orthosymétriques sont soit symétriques soit anti-symétriques. On remarque aussi que les formes sesquilinéaires orthosymétriques sont « assez proches » des formes hermitiennes : en définissant la matrice hermitienne $H = \sqrt{\beta}\, M$ on obtient $\langle x, y\rangle_{\text{H}} = \sqrt{\beta}\,\langle x, y\rangle_{\text{M}}$, pour tout $x, y \in \mathbf{C}^n$. Par conséquent, l'algèbre de Lie de $\langle\cdot,\cdot\rangle_{\text{H}}$ est identique à l'algèbre de Lie de $\langle\cdot,\cdot\rangle_{\text{M}}$:

$$\langle Lx, y\rangle_{\text{H}} = -\langle x, Ly\rangle_{\text{H}} \quad\Leftrightarrow\quad \sqrt{\beta}\,\langle Lx, y\rangle_{\text{M}} = -\sqrt{\beta}\,\langle x, Ly\rangle_{\text{M}} \quad\Leftrightarrow\quad \langle Lx, y\rangle_{\text{M}} = -\langle x, Ly\rangle_{\text{M}}\,.$$

De façon similaire, les algèbres de Jordan de $\langle\cdot,\cdot\rangle_{\text{H}}$ et de $\langle\cdot,\cdot\rangle_{\text{M}}$ sont aussi identiques. Par conséquent, un résultat établi pour les formes sesquilinéaires se transposent immédiatement en un résultat correspondant pour les formes sesquilinéaires orthosymétriques.

Ainsi, à un multiple scalaire près , il y a seulement trois types distincts de produits scalaires orthosymétriques : les formes bilinéaires symétriques et anti-symétriques et les formes sesquilinéaires hermitiennes.

12.2.2 Résultats auxiliaires

La caractérisation du conditionnement structuré et de l'erreur inverse structurée d'un système linéaire dépend de la solution du problème suivant : *étant donnée une classe de*

matrices structurées $\mathbb{S}$*, pour quels vecteurs* x, b *existe-t-il une matrice* $A \in \mathbb{S}$ *telle que* $Ax = b$ *?* Mackey, Mackey et Tisseur [136] ont donné une solution à ce problème quand $\mathbb{S}$ est une algèbre de Lie ou de Jordan d'un produit scalaire orthosymétrique.

Théorème 12.1 (Existence, [136, Thm. 3.2])**.** *Soit* $\langle \cdot, \cdot \rangle_M$ *un produit scalaire orthosymétrique, et soit* $\mathbb{S}$ *l'algèbre de Lie ou de Jordan correspondante. Alors pour toute paire de vecteurs* x*,* $b \in \mathbf{K}^n$ *avec* $x \neq 0$*, il existe* $A \in \mathbb{S}$ *tel que* $Ax = b$ *si et seulement si les conditions données dans la table ci-dessous sont satisfaites.*

Forme bilinéaire	$\mathbb{J}$	$\mathbb{L}$
symétrique	*toujours*	$\langle b, x \rangle_M = 0$
anti-symétrique	$\langle b, x \rangle_M = 0$	*toujours*
Forme sesquilinéaire	$\mathbb{J}$	$\mathbb{L}$
Hermitienne	$\langle b, x \rangle_M \in \mathbf{R}$	$\langle b, x \rangle_M \in i\mathbf{R}$

Le lemme suivant montre que quand $\mathbb{S}$ est l'algèbre de Lie ou de Jordan d'un produit scalaire orthosymétrique $\langle \cdot, \cdot \rangle_M$ sur $\mathbf{K}^n$, la multiplication à gauche par M envoie $\mathbb{S}$ sur les ensembles Skew($\mathbf{K}$) et Sym($\mathbf{K}$) pour les formes bilinéaires et à un scalaire près sur Herm($\mathbf{C}$) pour les formes sesquilinéaires où

$$\begin{aligned} \mathrm{Sym}(\mathbf{K}) &= \{A \in \mathbf{K}^{n\times n} : A^T = A\}, \\ \mathrm{Skew}(\mathbf{K}) &= \{A \in \mathbf{K}^{n\times n} : A^T = -A\}, \\ \mathrm{Herm}(\mathbf{C}) &= \{A \in \mathbf{C}^{n\times n} : A^* = A\} \end{aligned} \tag{12.2}$$

sont l'ensemble des matrices symétriques et anti-symétriques sur $\mathbf{K}^{n\times n}$ et des matrices Hermitiennes respectivement. C'est le résultat clé pour le traitement unifié du conditionnement et de l'erreur inverse structuré dans le cadre des algèbres de Lie et de Jordan.

Lemme 12.1. *Soit* $\mathbb{S}$ *une algèbre de Lie ou de Jordan associée à un produit scalaire orthosymétrique* $\langle \cdot, \cdot \rangle_M$*. Soit* $A \in \mathbb{S}$ *tel que* $A^\star = \delta A$ *avec* $\delta = \pm 1$*.*

- *Pour une forme bilinéaire sur* $\mathbf{K}^n$*,* ($\mathbf{K} = \mathbf{R}, \mathbf{C}$) *l'orthosymétrie signifie que* $M^T = \varepsilon M$ *avec* $\varepsilon = \pm 1$ *et alors*

$$M \cdot \mathbb{S} = \begin{cases} \mathrm{Sym}(\mathbf{K}) & \text{if } \delta = \varepsilon, \\ \mathrm{Skew}(\mathbf{K}) & \text{if } \delta \neq \varepsilon. \end{cases} \tag{12.3}$$

- *Pour une forme sesquilinéaire sur* $\mathbf{C}^n$*, l'orthosymétrie signifie que* $M^* = \beta M$ *pour un* $|\beta| = 1$ *et alors*

$$M \cdot \mathbb{S} = \begin{cases} \beta^{1/2}\mathrm{Herm}(\mathbf{C}) & si \ \ \delta = +1, \\ \imath\, \beta^{1/2}\mathrm{Herm}(\mathbf{C}) & si \ \ \delta = -1. \end{cases} \tag{12.4}$$

Démonstration. Pour les formes bilinéaires, on a

$$A \in \mathbb{S} \iff A^\star = M^{-1}A^TM = \delta A \iff MA = \delta A^TM.$$

En invoquant la propriété d'orthosymétrie, on a $(MA)^T = \delta(A^TM)^T = \varepsilon\delta(MA)$ ce qui prouve (12.3).

Pour les formes sesquilinéaires, un argument similaire au précédent montre que pour tout $A \in \mathbb{S}$, $\overline{\beta}^{1/2}MA$ est soit Hermitienne ou anti-Hermitienne. Mais l'ensemble des matrices anti-Hermitiennes est $\imath\,\mathrm{Herm}(\mathbf{C})$ si bien que (12.4) est vraie. ■

Le résultat suivant est une conséquence immédiate de la table 12.1 et du théorème 12.1.

Corollaire 12.1. *Soit* $\mathbb{S}$ *une des classes de matrices* $\mathrm{Sym}(\mathbf{K})$, $\mathrm{Skew}(\mathbf{K})$ *ou* $\mathrm{Herm}(\mathbf{C})$. *Alors pour toute paire de vecteurs* $x, b \in \mathbf{K}^n$ *avec* $x \neq 0$, *il existe* $A \in \mathbb{S}$ *tel que* $Ax = b$ *si et seulement si les conditions suivantes sont satisfaites :*

$\mathbb{S}$	*Condition*
$\mathrm{Sym}(\mathbf{K})$	*aucune*
$\mathrm{Skew}(\mathbf{K})$	$b^Tx = 0$
$\mathrm{Herm}(\mathbf{C})$	$b^*x \in \mathbf{R}$

Étant donnés x, b deux vecteurs de norme 1 (pour la norme 2), nous pouvons maintenant construire une matrice A appartenant à un des ensembles dans (12.2) telle que $Ax = b$ et $\|A\|_\nu \approx 1$, $\nu = 2, F$.

Il y a une exception pour les matrices complexes symétriques et la norme 2 pour laquelle nous ne sommes pas en mesure de construire une telle matrice A.

Lemme 12.2. *Soit* $\mathbb{S}$ *un des ensemble* $\mathrm{Sym}(\mathbf{K})$, $\mathrm{Skew}(\mathbf{K})$ *ou* $\mathrm{Herm}(\mathbf{C})$ *et* $x, b \in \mathbf{K}^n$ *deux vecteurs de norme* 1 *tels que la condition adéquate du corollaire 12.1 soit satisfaite.*

(i) Il existe une matrice $A \in \mathbb{S}$ *telle que* $Ax = b$ *et* $\|A\|_F = \sqrt{2 - \langle x, b\rangle^2} \leq \sqrt{2}$.

(ii) Si $\mathbb{S} \neq \mathrm{Sym}(\mathbf{C})$, *il existe une matrice* $A \in \mathbb{S}$ *telle que* $Ax = b$ *et* $\|A\|_2 = 1$.

(iii) Si $\mathbb{S} = \mathrm{Sym}(\mathbf{C})$, *il existe une matrice* $A \in \mathbb{S}$ *telle que* $Ax = \overline{x}$ *et* $\|A\|_2 = 1$.

Démonstration. (i) Si $\mathbb{S} = \mathrm{Sym}(\mathbf{K})$ alors $A = bx^* + \overline{x}b^T - (b^Tx)\overline{x}x^* \in \mathrm{Sym}(\mathbf{K})$ est telle que $Ax = b$. Il est facile de montrer que $\mathrm{trace}(A^*A) = 2 - |b^Tx|^2 \leq 2$ puisque $|b^Tx| \leq 1$.

Quand $\mathbb{S} = \mathrm{Skew}(\mathbf{K})$ le corollaire 12.1 nous dit que pour qu'il existe $A \in \mathbb{S}$ envoyant x sur b il faut $x^Tb = 0$. Un calcul simple montre alors que $A = bx^* - \overline{x}b^T \in \mathrm{Skew}(\mathbf{K})$ vérifie $Ax = b$ et $\|A\|_F = \sqrt{2}$.

Si $\mathbb{S} = \mathrm{Herm}(\mathbf{C})$, alors d'après le corollaire 12.1 nous devons avoir $b^*x \in \mathbf{R}$. Par conséquent $A = bx^* + xb^* - (b^*x)xx^* \in \mathrm{Herm}(\mathbf{C})$ vérifie $Ax = b$ et aussi $\mathrm{trace}(A^*A) = 2 - |b^*x|^2 \leq 2$ puisque $|b^*x| \leq 1$.

(ii) Si $\mathbb{S} = \mathrm{Sym}(\mathbf{R})$, la réflexion (de Householder) A envoyant le vecteur réel x sur le vecteur réel b est symétrique réelle et de norme 1 (pour la norme 2).

D'après le corollaire 12.1 il existe $A \in \mathrm{Skew}(\mathbf{K})$ envoyant x sur b si et seulement si $b^Tx = 0 \iff b^*\overline{x} = 0$. On en déduit qu'il existe une matrice unitaire Q satisfaisant $[\overline{x}, b] = Q[e_1, -e_2]$, où e_k dénote la k-ième colonne de la matrice identité. Par conséquent,

on vérifie facilement que $A = Q(e_1e_2^T - e_2e_1^T)Q^T$ satisfait $A^T = -A$, $Ax = b$ et $\|A\|_2 = 1$. Quand $x, b \in \mathbf{R}^n$, Q est orthogonale et alors A est anti-symétrique réelle.

Quand $\mathbb{S} = \mathrm{Herm}(\mathbf{C})$ avec $x \neq b$ il existe une unique réflexion unitaire A tel que $Ax = b$ [134, Thm. 8.6]. De plus $A = I + \gamma uu^*$, avec $u = b - x$ et $\gamma = 1/(u^*x)$. D'après le corollaire 12.1, on a $x^*b \in \mathbf{R}$ si bien que $u^*x = b^*x - 1 \in \mathbf{R} \setminus \{0\}$. Ainsi $A \in \mathrm{Herm}(\mathbf{C})$ et $\|A\|_2 = 1$ puisque A est unitaire.

(iii) Soit H une réflexion unitaire de Householder envoyant $\overline{x}$ sur $\|x\|_2 e_1 \in \mathbf{R}^n$ et posons $U = H^*$ si bien que $U^*\overline{x} = \|x\|_2 e_1 = \overline{U^T x} = U^T x$ puisque $\|x\|_2 e_1$ est réel. Soit $A = Ue_1e_1^TU^T$. Alors $A^T = A$ est symétrique complexe et

$$Ax = Ue_1e_1^T(U^Tx) = U(\|x\|_2 e_1) = UU^*\overline{x} = \overline{x}.$$

On a aussi $\|A\|_2 = \|e_1e_1^T\|_2 = 1$. ■

Une borne un peu plus faible pour $\|A\|_F$ avec A symétrique réelle est donnée dans [27, Thm 3] pour $\mathbf{K} = \mathbf{R}$ en utilisant une preuve constructive basée sur la factorisation QR de $[x, b]$. Le résultat pour la norme 2 et $A \in \mathrm{Skew}(\mathbf{R})$ est prouvé dans [172, Lem. 5.1].

12.3 Conditionnement structuré

12.3.1 Systèmes linéaires

Soit $\mathbb{S}$ une classe de matrices structurées (qui n'est pas nécessairement $\mathbb{J}$, $\mathbb{L}$ ou $\mathbb{G}$). On définit le conditionnement structuré du système linéaire $Ax = b$ avec $x \neq 0$ par

$$\begin{aligned}\mathrm{cond}(A, x; \mathbb{S}) \;=\; \lim_{\varepsilon \to 0} \sup \Big\{ \frac{\|\Delta x\|}{\varepsilon \|x\|} : (A + \Delta A)(x + \Delta x) = b + \Delta b, \qquad (12.5)\\ A + \Delta A \in \mathbb{S},\ \|\Delta A\| \leq \varepsilon \|A\|,\ \|\Delta b\| \leq \varepsilon \|b\| \Big\},\end{aligned}$$

où $\|\cdot\|$ est une norme matricielle arbitraire. Nous notons $\mathrm{cond}(A, x) \equiv \mathrm{cond}(A, x; \mathbf{K}^{n\times n})$ le conditionnement non structuré, où ici n est la dimension de A. Nous avons clairement

$$\mathrm{cond}(A, x; \mathbb{S}) \leq \mathrm{cond}(A, x).$$

Si cette inégalité n'est pas proche d'être atteinte alors il se peut que $\mathrm{cond}(A, x)$ surestime de manière importante le pire effet des perturbations structurées.

Définissons pour $A \in \mathbb{S}$ inversible et $x \in \mathbf{K}^n$ non nul,

$$\varphi(A, x; \mathbb{S}) = \lim_{\varepsilon \to 0} \sup \left\{ \frac{\|A^{-1}\Delta A x\|}{\varepsilon \|A\|} :\ \|\Delta A\| \leq \varepsilon \|A\|,\ A + \Delta A \in \mathbb{S} \right\}. \qquad (12.6)$$

Nous écrivons $\varphi(A, x) \equiv \varphi(A, x; \mathbf{K}^{n\times n})$ quand $A + \Delta A$ est non structurée et $\varphi_2(A, x)$ ou $\varphi_F(A, x)$ pour spécifier si nous utilisons la norme 2 ou la norme de Frobenius dans (12.6). De façon similaire, cond_2 et cond_F signifient que $\|\cdot\| = \|\cdot\|_2$ ou $\|\cdot\| = \|\cdot\|_F$ dans (12.5).

Le lemme suivant est utile pour la comparaison entre le conditionnement structuré et non structuré des systèmes linéaires.

Ce résultat est dû à Rump [172] et vaut pour toute classe de matrices structurées. Les preuves dans [172, Thm. 3.2 et Thm. 3.3] pour $\mathbf{K} = \mathbf{R}$ et la norme 2 s'étendent trivialement à $\mathbf{K} = \mathbf{C}$ et $\|\cdot\| = \|\cdot\|_2$ ou $\|\cdot\|_F$.

Lemme 12.3. *Soit $A \in \mathbf{K}^{n\times n}$ une matrice inversible et $x \in \mathbf{K}^n \setminus \{0\}$. Alors*

$$\frac{\varphi_\nu(A,x;\mathbb{S})}{\|x\|_2\|A^{-1}\|_2}\mathrm{cond}_\nu(A,x) \le \mathrm{cond}_\nu(A,x;\mathbb{S}) \le \mathrm{cond}_\nu(A,x), \quad \nu = 2, F.$$

et

$$\mathrm{cond}_\nu(A,x,\mathbb{S}) = \gamma\left(\varphi_\nu(A,x;\mathbb{S})\frac{\|A\|_\nu}{\|x\|_2} + \frac{\|A^{-1}\|_\nu\|b\|_2}{\|x\|_2}\right), \quad \nu = 2, F$$

où $\frac{1}{\sqrt{2}} \le \gamma \le 1$.

La difficulté pour obtenir une expression explicite pour $\varphi(A,x;\mathbb{S})$ dans (12.6) dépend fortement de la nature de $\mathbb{S}$ et de la norme matricielle $\|\cdot\|$ utilisée. Par exemple, quand $\|\cdot\|$ est la norme de Frobenius ou la norme 2 et que les perturbations sont non structurées (c.-à-d. $\mathbb{S} = \mathbf{K}^{n\times n}$), la borne supérieure dans (12.6) est atteinte pour $\Delta A = \varepsilon\|A\|_\nu yx^*/\|x\|_2$, où y est tel que $\|y\|_2 = 1$ et $\|A^{-1}y\|_2 = \|A^{-1}\|_2$. Cela nous donne la formule

$$\varphi_2(A,x) = \varphi_F(A,x) = \|A^{-1}\|_2\|x\|_2. \tag{12.7}$$

si bien que

$$\mathrm{cond}_\nu(A,x) = \|A^{-1}\|_2\|A\|_\nu + \frac{\|A^{-1}\|_2\|b\|_2}{\|x\|_2}, \quad \nu = 2, F.$$

Dans le reste de la section, nous nous intéressons à $\varphi(A,x;\mathbb{S})$ pour les matrices structurées dans $\mathbb{L}$, $\mathbb{J}$ ou $\mathbb{G}$ définies dans la section 12.2.1.

12.3.1.1 Algèbres de Lie et de Jordan

D.J. Higham [82] a prouvé que pour la structure symétrique réelle,

$$\mathrm{cond}_\nu(A,x;\mathrm{Sym}(\mathbf{R})) = \mathrm{cond}_\nu(A,x), \quad \nu = 2, F$$

et récemment Rump [172] a montré que cette égalité était encore valide en norme 2 pour les structures symétriques, persymétriques et anti-symétriques. Ce sont des exemples d'algèbres de Lie et de Jordan (voir la table 12.1). Nous étendons ces résultats à toutes les algèbres de Lie et de Jordan issues d'un produit scalaire orthosymétrique $\langle\cdot,\cdot\rangle_M$ avec M unitaire. Contrairement aux preuves de [172], nos preuves nécessitent de considérer chaque algèbre de manière individuelle.

Théorème 12.2. *Soit $\mathbb{S}$ une algèbre de Lie ou de Jordan associée à un produit scalaire orthosymétrique $\langle\cdot,\cdot\rangle_M$ sur $\mathbf{K}^n$ avec M unitaire. Étant donnés $A \in \mathbb{S}$ inversible et $x \neq 0$, on a*

$$\frac{\varphi_\nu(A,x)}{\sqrt{2}} \le \varphi_\nu(A,x;\mathbb{S}) \le \varphi_\nu(A,x), \quad \nu = 2, F,$$

et pour $\mathbb{S}$ *tel que* $M \cdot \mathbb{S} \neq \mathrm{Sym}(\mathbf{C})$,

$$\varphi_2(A, x; \mathbb{S}) = \varphi_2(A, x).$$

Démonstration. Puisque les algèbres de Lie et de Jordan sont des sous-espaces vectoriels de $\mathbf{K}^{n\times n}$, on peut écrire $\varphi(A, x, \mathbb{S})$ sous la forme

$$\varphi(A, x; \mathbb{S}) = \sup \left\{ \|A^{-1}\Delta A x\| : \ \|\Delta A\| \leq 1, \ \Delta A \in \mathbb{S} \right\}.$$

Pour un produit scalaire orthosymétrique $\langle \cdot, \cdot \rangle_{\mathrm{M}}$ avec M unitaire et pour toute norme unitairement invariante, on a

$$\varphi(A, x, \mathbb{S}) = \varphi(MA, x; M \cdot \mathbb{S}).$$

Puisqu'on a (12.7)

$$\varphi_\nu(MA, x; M \cdot \mathbb{S}) \leq \varphi_\nu(MA, x) = \|A^{-1}\|_2 \ \|x\|_2 = \varphi_\nu(A, x), \quad \nu = 2, F,$$

nous avons juste à montrer que $\mu\varphi_\nu(MA, x; M \cdot \mathbb{S}) \geq \|A^{-1}\|_2 \ \|x\|_2$ pour un $\mu \geq 1$. Pour cela, nous prouvons que pour un vecteur approprié $u \in \mathbf{K}^n$ de norme 1 (pour la norme 2) tel que $\|A^{-1}\|_2 = \|(MA)^{-1}\|_2 = \|(MA)^{-1}u\|_2$, il existe $E \in M \cdot \mathbb{S}$ vérifiant $Ex = \|x\|_2 u$ et $\|E\|_\nu \leq \mu$. Ainsi,

$$\|A^{-1}\|_2\|x\|_2 = \|(MA)^{-1}u\|_2\|x\|_2 = \|A^{-1}Ex\|_2 \leq \mu\varphi_\nu(MA, x; M \cdot \mathbb{S}).$$

Notons que d'après le lemme 12.1, $M \cdot \mathbb{S}$ est un des ensembles suivants $\mathrm{Sym}(\mathbf{K})$, $\mathrm{Skew}(\mathbf{K})$ ou $\mathrm{Herm}(\mathbf{C})$.

Sans perte de généralité, nous supposons maintenant que $\|x\|_2 = 1$. Si $M \cdot \mathbb{S} = \mathrm{Sym}(\mathbf{K})$, le lemme 12.2 nous dit alors qu'il existe $E \in \mathrm{Sym}(\mathbf{K})$ tel que $Ex = u$ et $\|E\|_\nu \leq \mu$ avec $\mu = 1$ si $\mathbf{K} = \mathbf{R}$, $\nu = 2$ et $\mu = \sqrt{2}$ sinon.

Si maintenant $M \cdot \mathbb{S} = \mathrm{Skew}(\mathbf{K})$ alors $A \in \mathbb{S}$ inversible implique que $(MA)^{-1} \in \mathrm{Skew}(\mathbf{K})$. Or on sait que les valeurs singulières d'une matrice inversible réelle ou complexe anti-symétrique ont des multiplicités paires [135, Thm. 8.4]. Ainsi il existe deux vecteurs orthogonaux u_1 et u_2 de $\mathbf{K}^n$ tels que $\|u_1\|_2 = \|u_2\|_2 = 1$ et $\|(MA)^{-1}u_1\|_2 = \|(MA)^{-1}u_2\|_2 = \|(MA)^{-1}\|_2 = \|A^{-1}\|_2$. Posons $u \in \mathrm{span}\{u_1, u_2\}$ vérifiant $u^T x = 0$ et $\|u\|_2 = 1$. Alors nécessairement, on a $\|(MA)^{-1}u\|_2 = \|(MA)^{-1}\|_2$. D'après le lemme 12.2, si $u^T x = 0$, alors il existe $E \in \mathrm{Skew}(\mathbf{K})$ tel que $Ex = u$ et $\|E\|_\nu = \mu$ avec $\mu = 1$ pour $\nu = 2$ et $\mu = \sqrt{2}$ pour $\nu = F$.

Pour finir si $M \cdot \mathbb{S} = \mathrm{Herm}(\mathbf{C})$, nous choisissons u telle que la contrainte $u^* x \in \mathbf{R}$ soit satisfaite. Alors d'après le lemme 12.2, il existe $E \in \mathrm{Herm}(\mathbf{C})$ tel que $Ex = u$ et $\|E\|_\nu \leq \mu$ avec $\mu = 1$ pour $\nu = 2$ et $\mu = \sqrt{2}$ pour $\nu = F$. ■

La condition $M \cdot \mathbb{S} \neq \mathrm{Sym}(\mathbf{C})$ dans le théorème 12.2 peut aussi se lire sous la forme : $\mathbb{S}$ n'est pas l'algèbre de Jordan d'une forme bilinéaire symétrique sur $\mathbf{C}^n$, ou $\mathbb{S}$ n'est pas l'algèbre de Lie d'une forme bilinéaire anti-symétrique sur $\mathbf{C}^n$. Des exemples de matrices structurées pour lesquelles $M \cdot \mathbb{S} \neq \mathrm{Sym}(\mathbf{C})$ incluent par exemple les matrices symétriques complexes, pseudo-symétriques complexes et J-symétriques complexes (voir le tableau 12.1).

Du lemme 12.3 et du théorème 12.2, on déduit le résultat suivant.

Théorème 12.3. *Soit $\mathbb{S}$ l'algèbre de Jordan ou de Lie associée à un produit scalaire orthosymétrique $\langle \cdot, \cdot \rangle_M$ sur $\mathbf{K}^n$ avec M unitaire. Soit $A \in \mathbb{S}$ une matrice inversible et $x \in \mathbf{K}^n$ avec $x \neq 0$. Alors*

$$\mathrm{cond}_\nu(A, x; \mathbb{S}) = \gamma \ \mathrm{cond}_\nu(A, x), \quad \nu = 2, F,$$

où $\gamma = 1$ pour la norme 2 et $M \cdot \mathbb{S} \neq \mathrm{Sym}(\mathbf{C})$ et $\dfrac{1}{\sqrt{2}} \leq \gamma \leq 1$ sinon.

Le théorème 12.3 montre que pour beaucoup d'algèbres de Lie et de Jordan, la contrainte $A + \Delta A \in \mathbb{S}$ n'a peu ou pas d'effets sur le conditionnement.

12.3.1.2 Groupes d'automorphismes

Quand $\mathbb{S}$ est une variété lisse (voir par exemple [13, 118, 126] pour une introduction aux variétés différentiables), le calcul de la borne supérieure (12.6) se simplifie en un problème d'optimisation sous contraintes linéaires.

Lemme 12.4. *Soit $\mathbb{S}$ une variété lisse réelle ou complexe. Alors pour une matrice $A \in \mathbb{S}$ inversible, on a*

$$\varphi(A, x; \mathbb{S}) = \sup \left\{ \|A^{-1}Ex\| : \ \|E\| = 1, \ E \in T_A\mathbb{S} \right\},$$

où $T_A\mathbb{S}$ est l'espace tangent en A à la variété $\mathbb{S}$.

Démonstration. Soit $E \in T_A\mathbb{S}$ avec $\|E\| = 1$. Par définition de l'espace tangent, il existe une courbe lisse $g_E : (-\varepsilon, \varepsilon) \to \mathbf{K}^{n \times n}$ vérifiant $g_E(0) = 0$, $g_E'(0) = E$ et $A + g_E(t) \in \mathbb{S}$ pour tout t. Nous avons

$$\lim_{t \to 0} \frac{g_E(t)}{\|g_E(t)\|} = \lim_{t \to 0} \frac{Et + \mathcal{O}(|t|^2)}{\|Et + \mathcal{O}(|t|^2)\|} = E.$$

Ainsi,

$$\lim_{t \to 0} \frac{\|A^{-1} g_E(t) x\|}{\|g_E(t)\|} = \|A^{-1}Ex\|.$$

Cela implique que $\varphi(A, x; \mathbb{S}) \geq \sup \{\|A^{-1}Ex\| : \ \|E\| = 1, \ E \in T_A\mathbb{S}\}$. L'égalité vient alors du fait que les courbes g_E forment un recouvrement d'un voisinage ouvert de $A \in \mathbb{S}$. ■

Un groupe d'automorphismes $\mathbb{G}$ forme une variété lisse (et donc un groupe de Lie). La différentielle de la fonction

$$F(A) = \begin{cases} A^T M A - M \text{ (formes bilinéaires)} \\ A^* M A - M \text{ (formes sesquilinéaires)} \end{cases}$$

en $A \in \mathbf{K}^{n \times n}$ est donc l'application linéaire

$$J_A(X) = \begin{cases} A^T M X + X^T M A \text{ (formes bilinéaires)} \\ A^* M X + X^* M A \text{ (formes sesquilinéaires).} \end{cases}$$

L'espace tangent $T_A\mathbb{G}$ en $A \in \mathbb{G}$ coïncide avec le noyau de cette différentielle,

$$T_A\mathbb{G} = \{X : \ J_A(X) = 0\} = \{AH : \ H^\star = -H\} = A \cdot \mathbb{L}, \tag{12.8}$$

où $\mathbb{L}$ est l'algèbre de Lie associée à $\langle \cdot, \cdot \rangle_{\mathrm{M}}$.

De la même façon que dans [83] et [115], on peut obtenir une expression explicite de $\kappa(A, \lambda; \mathbb{S})$ si l'on suppose que la norme $\| \cdot \|$ utilisée est la norme de Frobenius $\| \cdot \|_F$. Écrivons que

$$Ex = (I_n \otimes x^T)\mathrm{vec}(E^T),$$

où $\otimes$ est le produit de Kronecker et vec l'opérateur qui empile les colonnes de la matrice en un seul vecteur [66, p. 180]. On remarque facilement que $T_A\mathbb{G} = A \cdot \mathbb{L}$ est un espace vectoriel de dimension $m \leq n^2$. Par conséquent, il existe une matrice B de taille $n^2 \times m$ telle que pour tout $E \in T_A\mathbb{G}$, il existe un paramètre vectoriel p tel que

$$\mathrm{vec}(E) = Bp, \quad \|E\|_F = \|p\|_2. \tag{12.9}$$

Toute matrice B satisfaisant ces propriétés est appelée une *matrice modèle* pour $T_A\mathbb{S}$. Puisque l'algèbre de Lie $\mathbb{L}$ dans (12.8) est indépendante de A, il est souvent plus simple de construire explicitement une matrice modèle L telle que pour tout $H \in \mathbb{L}$, il existe un paramètre vectoriel q avec

$$\mathrm{vec}(H) = Lq, \quad \|H\|_F = \|q\|_2.$$

Pour obtenir une matrice modèle B pour $A \cdot \mathbb{L}$ dans le sens de (12.9), on calcule une décomposition QR $(I \otimes A)L = BR$, où les colonnes de B forment une base orthonormale de l'espace engendré par les colonnes de L et R est une matrice triangulaire supérieure. Alors,

$$\mathrm{vec}(AH) = (I \otimes A)\mathrm{vec}(H) = (I \otimes A)Lq = Bp,$$

où $p = Rq$ et $\|AH\|_F = \|\mathrm{vec}(AH)\|_2 = \|p\|_2$. Ainsi, d'après le lemme 12.4, on a

$$\varphi_F(A, x; \mathbb{G}) = \sup\{\|A^{-1}(I_n \otimes x^T)Bp\|_2 : \ \|p\|_2 = 1, p \in \mathbf{K}^n\}.$$

C'est pourquoi, si $\mathbf{K} = \mathbf{R}$ et B, x sont réels, alors

$$\varphi_F(A, x; \mathbb{G}) = \|A^{-1}(I_n \otimes x^T)B\|_2. \tag{12.10}$$

Autrement, si $\mathbf{K} = \mathbf{R}$ (p est réel) et B ou x sont complexes, alors

$$\varphi_F(A, x; \mathbb{G}) = \left\| \begin{array}{c} \mathrm{Re}(A^{-1}(I_n \otimes x^T)B) \\ \mathrm{Im}(A^{-1}(I_n \otimes x^T)B) \end{array} \right\|_2. \tag{12.11}$$

Ainsi d'après le lemme 12.3, nous avons, à un petit facteur près, une expression directement calculable pour $\mathrm{cond}_F(A, x; \mathbb{G})$. La construction d'une matrice modèle B est rendue plus facile quand $\langle \cdot, \cdot \rangle_{\mathrm{M}}$ est orthosymétrique (voir [115, Sec. 5.1]).

Un des inconvénients de la formule explicite pour $\varphi_F(A, x; \mathbb{G})$ dans (12.10) ou (12.11) est qu'il est difficile de la comparer avec $\varphi_F(A, x) = \|A^{-1}\|_2\|x\|_2$. Cependant, quand $\langle \cdot, \cdot \rangle_{\mathrm{M}}$ est orthosymétrique avec M unitaire, on peut obtenir une borne inférieure pour $\varphi(A, x; \mathbb{G})$ comme le montre le théorème suivant.

Théorème 12.4. *Soit $\mathbb{G}$ un groupe d'automorphismes associé à un produit scalaire orthosymétrique $\langle\cdot,\cdot\rangle_M$ sur $\mathbf{K}^n$ avec M unitaire. Soient $A \in \mathbb{G}$ et $x \in \mathbf{K}^n$ avec $x \neq 0$. Alors*

$$\gamma \frac{\varphi_\nu(A,x)}{\|A\|_2\|A^{-1}\|_2} \leq \varphi_\nu(A,x;\mathbb{G}) \leq \varphi_\nu(A,x),$$

avec $\gamma = 1$ si $\nu = 2$ et $\gamma = 1/\sqrt{2}$ si $\nu = F$.

Démonstration. Soit $\mathbb{L}$ l'algèbre de Lie associée à $\langle\cdot,\cdot\rangle_M$. D'après le lemme 12.4 et (12.8), on a

$$\begin{aligned}\varphi_\nu(A,x;\mathbb{S}) &= \sup\left\{\|A^{-1}Ex\|_2 : \ \|E\|_\nu = 1, \ E \in T_A\mathbb{G}\right\},\\ &= \sup\left\{\|Hx\|_2 : \ \|AH\|_\nu = 1, \ H \in \mathbb{L}\right\},\\ &= \sup\left\{\|\widetilde{H}x\|_2 : \ \|AM^*\widetilde{H}\|_\nu = 1, \ \widetilde{H} \in \widetilde{\mathbb{L}}\right\},\end{aligned}$$

où $\widetilde{H} = MH$ et d'après le lemme 12.1, $\widetilde{\mathbb{L}} = M\cdot\mathbb{S}$ est un des ensembles $\mathrm{Sym}(\mathbf{K})$, $\mathrm{Skew}(\mathbf{K})$ ou $\mathrm{Herm}(\mathbf{C})$. Soit $u \in \mathbf{K}^n$ de norme 1 (pour la norme $\|\cdot\|_2$) et satisfaisant la contrainte $x^Tu = 0$ si $\widetilde{\mathbb{L}} = \mathrm{Skew}(\mathbf{K})$ et $u^*x \in \mathbf{R}$ si $\widetilde{\mathbb{L}} = \mathrm{Herm}(\mathbf{C})$. D'après la première partie du lemme 12.2, il existe $S \in \widetilde{\mathbb{L}}$ tel que $S(x/\|x\|) = u$ avec $\|S\|_2 \leq \|S\|_F \leq \sqrt{2}$. Soit $\widetilde{H} = \xi S \in \widetilde{\mathbb{L}}$ avec $\xi > 0$ tel que $\|AM^*\widetilde{H}\|_\nu = 1$. Cela implique $1 = \xi\|AM^*S\|_\nu \leq \xi\|A\|_2\|S\|_\nu \leq \xi\sqrt{2}\|A\|_2$ si bien que d'après (12.7),

$$\varphi_\nu(A,x;\mathbb{G}) \geq \|\widetilde{H}x\|_2 = \xi\|x\|_2 \geq \frac{\|x\|_2}{\sqrt{2}\|A\|_2} = \frac{\varphi_\nu(A,x)}{\sqrt{2}\|A\|_2\|A^{-1}\|_2}.$$

■

Le lemme 12.3 et le théorème 12.4 entraînent le résultat suivant.

Théorème 12.5. *Soit $\mathbb{G}$ un groupe d'automorphismes associé à un produit scalaire orthosymétrique $\langle\cdot,\cdot\rangle_M$ sur $\mathbf{K}^n$ avec M unitaire. Soient $A \in \mathbb{G}$ et $x \in \mathbf{K}^n$ avec $x \neq 0$. Alors*

$$\gamma \frac{\mathrm{cond}_\nu(A,x)}{\|A\|_2\|A^{-1}\|_2} \leq \mathrm{cond}_\nu(A,x;\mathbb{G}) \leq \mathrm{cond}_\nu(A,x),$$

où $\gamma = 1/\sqrt{2}$ si $\nu = 2$ et $\gamma = 1/2$ si $\nu = F$.

Le théorème 12.5 montre que quand A est bien conditionnée, les nombres de conditionnement structurés et non-structurés pour le système $Ax = b$ sont quasiment les mêmes. Cependant, quand A est mal conditionnée, c.-à-d. $\kappa_2(A) \gg 1$, la borne inférieure n'est pas fine.

12.3.2 Inversion de matrices

Soit $\mathbb{S}$ une classe de matrices structurées. Le conditionnement structuré pour le calcul de l'inverse d'une matrice est défini par

$$\kappa(A;\mathbb{S}) = \lim_{\varepsilon\to 0}\sup\left\{\frac{\|(A+\Delta A)^{-1} - A^{-1}\|}{\varepsilon\|A^{-1}\|} : \ A+\Delta A \in \mathbb{S}, \ \|\Delta A\| \leq \varepsilon\|A\|\right\}. \qquad (12.12)$$

Quand ΔA n'est pas structurée, on écrit $\kappa(A) \equiv \kappa(A; \mathbf{K}^{n\times n})$. Pour la norme 2 et la norme de Frobenius, $\kappa(A)$ est caractérisé par

$$\kappa_2(A) = \|A\|_2\|A^{-1}\|_2, \qquad \kappa_F(A) = \frac{\|A\|_F\|A^{-1}\|_2^2}{\|A^{-1}\|_F}. \tag{12.13}$$

(Voir [87, Th. 6.4] pour $\nu = 2$ et [82] pour $\nu = F$.)

Là encore, quand $\mathbb{S}$ est une variété lisse le supremum (12.6) se simplifie en un problème de minimisation avec contraintes linéaires.

Lemme 12.5. *Soit $\mathbb{S}$ une variété lisse réelle ou complexe. Alors pour $A \in \mathbb{S}$ inversible, on a*

$$\kappa(A; \mathbb{S}) = \frac{\|A\|}{\|A^{-1}\|} \sup\{\|A^{-1}\Delta A A^{-1}\| : \ \|\Delta A\| = 1, \ \Delta A \in T_A\mathbb{S}\}, \tag{12.14}$$

où $T_A\mathbb{S}$ est l'espace tangent en A.

Démonstration. Soit $E \in T_A\mathbb{S}$ avec $\|E\| = 1$. Comme dans la preuve du lemme 12.4, il existe une courbe lisse $g_E : (-\varepsilon, \varepsilon) \to \mathbf{K}^{n\times n}$ vérifiant $g_E(0) = 0$, $g_E'(0) = E$ et $A + g_E(t) \in \mathbb{S}$ pour tout t. On a aussi

$$\lim_{t\to 0} \frac{g_E(t)}{\|g_E(t)\|} = E.$$

Au vue de l'expansion

$$(A + \Delta A)^{-1} - A^{-1} = -A^{-1}\Delta A A^{-1} + \mathcal{O}(\|\Delta A\|^2),$$

on a

$$\lim_{t\to 0} \frac{\|(A + g_E(t))^{-1} - A^{-1}\|}{\|g_E(t)\|} = \lim_{t\to 0} \frac{A^{-1}g_E(t)A^{-1} + \mathcal{O}(\|g_E(t)\|^2)}{\|g_E(t)\|} = \|A^{-1}EA^{-1}\|.$$

Cela implique que $\kappa(A; \mathbb{S}) \geqslant \eta$, où η dénote la partie droite de l'équation (12.14). L'égalité a lieu car les courbes g_E forme un recouvrement d'un voisinage ouvert de $A \in \mathbb{S}$. ∎

Quand $\mathbb{S}$ est une algèbre de Lie ou de Jordan associée à un produit scalaire, l'espace tangent à $\mathbb{S}$ est $\mathbb{S}$ lui-même : $T_A\mathbb{S} = \mathbb{S}$. Si on se restreint à un produit scalaire $\langle\cdot,\cdot\rangle_M$ orthosymétrique avec M unitaire, on a alors égalité entre le conditionnement structuré et non structuré pour la norme 2 et la norme de Frobenius.

Théorème 12.6. *Soit $\mathbb{S}$ une algèbre de Lie ou de Jordan associée à un produit scalaire orthosymétrique $\langle\cdot,\cdot\rangle_M$ sur $\mathbf{K}^n$ avec M unitaire. Pour une matrice $A \in \mathbb{S}$ inversible, on a*

$$\kappa_\nu(A; \mathbb{S}) = \kappa_\nu(A), \quad \nu = 2, F.$$

Démonstration. D'après l'inégalité $\|ABC\|_\nu \leqslant \|A\|_2\|B\|_\nu\|C\|_2$, on a

$$\kappa_\nu(A; \mathbb{S}) \leqslant \|A\|_\nu\|A^{-1}\|_2^2/\|A^{-1}\|_\nu = \kappa_\nu(A).$$

Puisque M est unitaire, on a

$$\begin{aligned}\kappa_\nu(A;\mathbb{S}) &= \frac{\|\widetilde{A}\|_\nu}{\|\widetilde{A}^{-1}\|_\nu}\sup\{\|\widetilde{A}^{-1}\widetilde{E}\widetilde{A}^{-1}\|_\nu : \|\widetilde{E}\| = 1,\ \widetilde{E} \in M\cdot\mathbb{S}\}\\ &= \kappa_\nu(\widetilde{A},\widetilde{\mathbb{S}}),\end{aligned}$$

avec $\widetilde{\mathbb{S}} = M\cdot\mathbb{S}$ et $\widetilde{A} = MA \in \widetilde{\mathbb{S}}$. D'après le lemme 12.1, l'orthosymétrie de $\langle\cdot,\cdot\rangle_M$ implique que $\widetilde{\mathbb{S}}$ est un des ensembles Sym($\mathbf{K}$), Skew($\mathbf{K}$) et Herm($\mathbf{C}$) si bien que nous devons juste prouver que $\kappa_\nu(A;\mathbb{S}) \geqslant \|A\|_\nu\|A^{-1}\|_2^2/\|A^{-1}\|_\nu$ pour chacune de ces trois classes.

- $\widetilde{\mathbb{S}} = \mathrm{Sym}(\mathbf{K})$. Considérons la factorisation de Takagi de $\widetilde{A}^{-1} \in \mathrm{Sym}(\mathbf{K})$, $\widetilde{A}^{-1} = U\Sigma U^T$, où U est unitaire et $\Sigma = \mathrm{diag}(\sigma_1,\sigma_2,\ldots,\sigma_n)$ avec $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_n \geqslant 0$ [99, Cor. 4.4.4]. Quand $\mathbf{K} = \mathbf{R}$, U est orthonormée et $\widetilde{A}^{-1} = U\Sigma U^T$ est la décomposition en valeurs singulières de $\widetilde{A}^{-1}$. Posons $E = \overline{U}e_1e_1^TU^*$.
- $\widetilde{\mathbb{S}} = \mathrm{Skew}(\mathbf{K})$. On considère l'analogue anti-symétrique de la factorisation de Takagi (Problèmes 25 et 26 dans [99, Sec. 4.4]). Puisque $\widetilde{A}^{-1} \in \mathrm{Skew}(\mathbf{K})$, il existe une matrice unitaire U telle que $\widetilde{A}^{-1} = UDU^T$ avec $D = D_1 \oplus \cdots \oplus D_{n/2}$ et

$$D_j = \begin{bmatrix} 0 & z_j \\ -z_j & 0 \end{bmatrix}, \quad 0 \neq z_j \in \mathbf{C}, \quad j = 1:n/2.$$

  Quand $\mathbf{K} = \mathbf{R}$, U est orthogonale. Supposons que les z_j sont ordonnés de façon à ce que $\sigma_{\max}^{-1} = |z_1| \geqslant |z_2| \geqslant \cdots \geqslant |z_k|$. Posons alors $E = \gamma\overline{U}(e_1e_2^T - e_2e_1^T)U^*$ avec $\gamma = 1$ si $\nu = 2$ et $\gamma = 1/\sqrt{2}$ si $\nu = F$.
- $\widetilde{\mathbb{S}} = \mathrm{Herm}(\mathbf{K})$.Posons $E = Ue_1e_1U^*$ où U est le facteur unitaire dans la décomposition en valeurs singulières de A^{-1}.

Dans chaque cas, nous avons $E \in \widetilde{\mathbb{S}}$, $\|E\|_\nu = 1$ et $\|\widetilde{A}^{-1}E\widetilde{A}^{-1}\|_\nu = \sigma_{\max}(A^{-1})^2 = \|A^{-1}\|_2^2$ si bien que d'après (12.14), $\kappa(\widetilde{A},\widetilde{\mathbb{S}}) \geqslant \|A\|_\nu\|A^{-1}\|_2^2/\|A^{-1}\|_\nu$. ■

Quand $\mathbb{S}$ est un groupe d'automorphismes $\mathbb{G}$ associé à un produit scalaire, on peut donner un expression calculable pour $\kappa_{\mathrm{F}}(A;\mathbb{G})$. Soit B une matrice modèle pour $T_A\mathbb{G} = A\cdot\mathbb{L}$ au sens de (12.9). On a alors

$$\|A^{-1}EA^{-1}\|_F = \|\mathrm{vec}(A^{-1}EA^{-1})\|_2 = \|(A^T\otimes A)^{-1}\mathrm{vec}(E)\|_2 = \|(A^T\otimes A)^{-1}Bp\|_2$$

si bien que d'après (12.14),

$$\kappa_{\mathrm{F}}(A;\mathbb{G}) = \frac{\|A\|_F}{\|A^{-1}\|_F}\|(A^T\otimes A)^{-1}B\|_2. \tag{12.15}$$

On a $\kappa_2(A) \geqslant 1$. Il est en de même pour $\kappa_2(A,\mathbb{G})$ quand $\mathbb{G}$ est un groupe d'automorphismes associé à un produit scalaire orthosymétrique $\langle\cdot,\cdot\rangle_M$ avec M unitaire. En effet, soient $u,v \in \mathbf{K}^n$ tels que $A^{-1}u = \|A^{-1}\|_2 v$ avec $\|u\|_2 = \|v\|_2 = 1$. Alors, d'après (12.14)

et (12.8),

$$\begin{aligned}\kappa_2(A;\mathbb{G}) &= \frac{\|A\|_2}{\|A^{-1}\|_2}\sup\{\|HA^{-1}\|_2 : \|AH\|_2 = 1,\ H \in \mathbb{L}\},\\ &\geqslant \frac{\|A\|_2}{\|A^{-1}\|_2}\|A^{-1}\|_2\sup\{\|Hv\|_2 : \|AH\|_2 = 1,\ H \in \mathbb{L}\},\\ &= \|A\|_2\varphi_2(A,v;\mathbb{G}).\end{aligned}$$

De plus d'après le théorème 12.4, $\varphi_2(A,v;\mathbb{G}) \geqslant 1/\|A\|_2$ si bien que $1 \leqslant \kappa_2(A;\mathbb{G}) \leqslant \kappa_2(A)$. Il est clair que si A est bien conditionné, c.-à-d. $\kappa_2(A)$ est petit, alors $\kappa_2(A;\mathbb{G}) \approx \kappa_2(A)$.

12.3.3 Distance à la singularité

La *distance structurée à la singularité* d'une matrice est définie par

$$\delta(A;\mathbb{S}) = \min\Big\{\varepsilon :\ \|\Delta A\| \leq \varepsilon\|A\|,\ \ A + \Delta A \text{ singulière},\ \ A + \Delta A \in \mathbb{S}\Big\}. \tag{12.16}$$

D. J. Higham [82] a montré que pour des perturbations symétriques,

$$\delta_\nu(A;\mathrm{Sym}(\mathbf{R})) = \delta_\nu(A), \quad \nu = 2, F,$$

si bien que la contrainte $\Delta A = \Delta A^T$ n'a aucun effet sur la distance. C'est un cas particulier d'un résultat plus général que voici.

Théorème 12.7. *Soit $\mathbb{S}$ une algèbre de Lie ou de Jordan associée à un produit scalaire orthosymétrique $\langle\cdot,\cdot\rangle_M$ sur $\mathbf{K}^n$ avec M unitaire. Soit $A \in \mathbb{S}$ une matrice inversible. Alors on a*

$$\delta_\nu(A;\mathbb{S}) = \gamma_\nu\delta_\nu(A) = \gamma_\nu\,\frac{1}{\|A\|_\nu\|A^{-1}\|_2}, \quad \nu = 2, F,$$

avec $\gamma_2 = 1$ et $1 \leq \gamma_F \leq \sqrt{2}$ si $M\cdot\mathbb{S} = \mathrm{Skew}(\mathbf{K})$ et $\gamma_F = 1$ sinon.

Démonstration. La distance relative à la singularité est égale à l'inverse du conditionnement $\kappa(A)$ [87, p.111]. Il est clair que $\delta_\nu(A;\mathbb{S}) \geqslant \delta_\nu(A)$. Pour un produit scalaire orthosymétrique $\langle\cdot,\cdot\rangle_M$ sur $\mathbf{K}^n$ avec M unitaire, nous avons

$$\delta_\nu(A;\mathbb{S}) = \delta_\nu(\widetilde{A},\widetilde{\mathbb{L}}),$$

avec $\widetilde{A} = MA$ et $\widetilde{\mathbb{L}} = M\cdot\mathbb{L}$ est une des trois structures $\mathrm{Sym}(\mathbf{K})$, $\mathrm{Skew}(\mathbf{K})$ et $\mathrm{Herm}(\mathbf{K})$. Sans perte de généralité, on peut supposer que $\|\widetilde{A}\|_\nu = 1$. Puisque $(\widetilde{A} + \Delta\widetilde{A})^{-1} = (I + \widetilde{A}^{-1}\Delta\widetilde{A})^{-1}\widetilde{A}^{-1}$ il nous suffit juste d'exhiber $\Delta\widetilde{A} \in \widetilde{\mathbb{S}}$ tel que $I + \widetilde{A}^{-1}\Delta\widetilde{A}$ soit singulière et $\|\Delta\widetilde{A}\|_\nu \leqslant \gamma/\|A^{-1}\|_2$. Cela est obtenu en prenant $\Delta\widetilde{A} = -E/\sigma_{\max}(A^{-1})$ où E est la matrice définie dans la preuve du théorème 12.6. ∎

On n'a pas besoin de considérer le cas $\mathbb{S} = \mathbb{G}$ car tout élément $A \in \mathbb{G}$ est inversible.

12.4 Erreur inverse structurée

Quand on résout un système linéaire $Ax = b$ avec $A \in \mathbb{S} \subset \mathbf{K}^{n\times n}$, il est intéressant de savoir si la solution calculée y est la solution d'un système proche structuré : il s'agit par exemple de savoir si l'erreur inverse structurée

$$\begin{aligned} \mu(y, r, \mathbb{S}) &= \min\{\|\Delta A\| : \ (A + \Delta A)y = b, \ A + \Delta A \in \mathbb{S}\} \\ &= \min\{\|\Delta A\| : \ \Delta A y = r, \ A + \Delta A \in \mathbb{S}\}, \end{aligned} \tag{12.17}$$

est relativement petite. Ici r est le résidu $r = b - Ay$. Notons que dans (12.17), seule la matrice A est perturbée. Nous étudierons un peu plus tard le cas où A et b sont perturbés.

12.4.1 Cas particulier

Comme nous le verrons dans la prochaine sous-section, étudier le cas particulier de l'erreur inverse $\mu_\nu(y, r; \mathbb{S})$ est essentiel avant de considérer le cas général où à la fois A et b sont perturbés. L'erreur inverse structurée pour une paire vecteur propre/valeur propre (x, λ) de $A \in \mathbb{S}$ est définie par

$$\widetilde{\mu}_\nu(x, \lambda; \mathbb{S}) = \min\{\|E\|_\nu : \ (A + E)x = \lambda x, \ E \in \mathbb{S}\}, \quad \nu = 2, F$$

et est égale à $\mu_\nu(x, r_\lambda; \mathbb{S})$ avec $r_\lambda = (\lambda I - A)x$. Ainsi, une caractérisation explicite de $\mu_\nu(y, r; \mathbb{S})$ entraîne automatiquement une caractérisation explicite pour l'erreur inverse du problème de valeur propre/vecteur propre.

Notons E_{opt} la solution optimale définie par $\|E_{\text{opt}}\|_\nu = \mu_\nu(y, r; \mathbb{S})$. Pour des perturbations non-structurées, on a

$$\mu_\nu(y, r) \equiv \mu_\nu(y, r, \mathbf{K}^{n\times n}) = \frac{\|r\|_2}{\|y\|_2}, \quad \nu = 2, F$$

et que $E_{\text{opt}} = ry^*/(y^*y)$. Pour les matrices réelles symétriques et pour la norme de Frobenius, nous savons que [27], [85],

$$\mu_F(y, r, \text{Sym}(\mathbf{R})) = \frac{1}{\|y\|_2}\sqrt{2\|r\|_2^2 - \frac{(y^T r)^2}{\|y\|_2^2}}$$

et donc

$$\mu_F(y, r) \leq \mu_F(y, r, \text{Sym}(\mathbf{R})) \leq \sqrt{2}\mu_F(y, r).$$

Par conséquent, forcer les perturbations à rester symétriques n'a très peu d'effets sur la norme de la perturbation optimale. Nous montrons maintenant que ce résultat se généralise aux matrices structurées considérées dans ce chapitre.

Quand $\mathbb{S}$ est une algèbre de Lie ou de Jordan associée à un produit scalaire orthosymétrique $\langle \cdot, \cdot \rangle_M$ avec M unitaire, il est facile de montrer que

$$\mu_\nu(y, r, \mathbb{S}) = \mu_\nu(y, \widetilde{r}, \widetilde{\mathbb{S}}), \tag{12.18}$$

où $\widetilde{\mathbb{S}} = M \cdot \mathbb{S}$, $\widetilde{r} = Mr$ et $\|\widetilde{r}\|_2 = \|r\|_2$. Par conséquent, d'après (12.3) et (12.4) nous n'avons qu'à étudier $\mu_\nu(y, r, \mathbb{S})$ pour $\mathbb{S} = \text{Sym}(\mathbf{K})$, $\text{Skew}(\mathbf{K})$ et $\text{Herm}(\mathbf{C})$.

Lemme 12.6. *Soient $y, r \in \mathbf{K}^n$ avec $y \neq 0$.*

– *Pour $\mathbb{S} = \mathrm{Sym}(\mathbf{K}) = \{A \in \mathbf{K}^{n\times n} : \ A^T = A\}$,*

$$\|E_{\mathrm{opt}}\|_F = \frac{1}{\|y\|_2}\sqrt{2\|r\|_2^2 - \frac{(y^T r)^2}{\|y\|_2^2}} \quad \textit{avec} \quad E_{\mathrm{opt}} = \frac{ry^* + \overline{y}r^T}{y^*y} - (r^T y)\frac{\overline{y}y^T}{(y^*y)^2}.$$

– *Pour $\mathbb{S} = \mathrm{Skew}(\mathbf{K}) = \{A \in \mathbf{K}^{n\times n} : \ A^T = -A\}$ et si $r^T y = 0$ alors*

$$\|E_{\mathrm{opt}}\|_F = \sqrt{2}\frac{\|r\|_2}{\|y\|_2} \quad \textit{avec} \quad E_{\mathrm{opt}} = \frac{ry^* - \overline{y}r^T}{y^*y}.$$

– *Pour $\mathbb{S} = \mathrm{Herm}(\mathbf{C}) = \{A \in \mathbf{C}^{n\times n} : \ A^* = A\}$ et si $r^* y \in \mathbf{R}$ alors*

$$\|E_{\mathrm{opt}}\|_F = \frac{1}{\|y\|_2}\sqrt{2\|r\|_2^2 - \frac{(y^* r)^2}{\|y\|_2^2}} \quad \textit{avec} \quad E_{\mathrm{opt}} = \frac{ry^* + yr^*}{y^*y} - (r^* y)\frac{yy^*}{(y^*y)^2}.$$

Démonstration. Soit $\mathcal{A}(\mathbb{S}) = \{A \in \mathbb{S} : \ Ay = r\}$. Sous les conditions du corollaire 12.1, on a $\mathcal{A}(\mathbb{S}) \neq \emptyset$ et [136, Thm. 3.5]

$$w = \begin{cases} \overline{y}/y^*y \text{ for } \mathbb{S} = \mathrm{Sym}(\mathbf{K}) \text{ or } \mathrm{Skew}(\mathbf{K}), \\ y/y^*y \text{ for } \mathbb{S} = \mathrm{Herm}(\mathbf{C}), \end{cases}$$

si bien que $w^T y = 1$ entraîne

$$\begin{aligned}
\mathcal{A}(\mathrm{Sym}(\mathbf{K})) &= \left\{ \frac{ry^* + \overline{y}r^T}{y^*y} - (r^T y)\frac{\overline{y}y^T}{(y^*y)^2} + (I - \frac{\overline{y}y^T}{y^*y})S(I - \frac{yy^*}{y^*y}),\ S \in \mathrm{Sym}(\mathbf{K}) \right\}, \\
\mathcal{A}(\mathrm{Skew}(\mathbf{R})) &= \left\{ \frac{ry^* - \overline{y}r^T}{y^*y} + (I - \frac{\overline{y}y^T}{y^*y})S(I - \frac{yy^*}{y^*y}),\ S \in \mathrm{Skew}(\mathbf{K}) \right\}, \\
\mathcal{A}(\mathrm{Herm}(\mathbf{C})) &= \left\{ \frac{ry^* + yr^*}{y^*y} - (r^* y)\frac{yy^*}{(y^*y)^2} + (I - \frac{yy^*}{y^*y})S(I - \frac{yy^*}{y^*y}),\ S \in \mathrm{Herm}(\mathbf{C}) \right\}.
\end{aligned}$$

Soit $E \in \mathcal{A}(\mathbb{S})$ avec $\mathbb{S} = \mathrm{Sym}(\mathbf{K})$, $\mathrm{Skew}(\mathbf{K})$ ou $\mathrm{Herm}(\mathbf{C})$. Il est clair que $E_{\mathrm{opt}} \in \mathcal{A}(\mathbb{S})$ et $\|Ey\|_2 = \|E_{\mathrm{opt}}y\|_2 = \|r\|_2$ pour tout E. Soit $v \in \mathbf{K}^n$ tel que $y^*v = 0$. Alors pour $\mathbb{S} = \mathrm{Sym}(\mathbf{K})$ ou $\mathrm{Skew}(\mathbf{K})$

$$\|E_{\mathrm{opt}}v\|_2^2 = (r^T v)^2, \qquad \|Ev\|_2^2 = (r^T v)^2 + \|(I - yy^T)Sv\|_2^2,$$

et quand $\mathbb{S} = \mathrm{Herm}(\mathbf{C})$ nous avons des expressions similaires pour $\|E_{\mathrm{opt}}v\|_2^2$ et $\|Ev\|_2^2$ avec ^{T} replacé par *. On en déduit que $\|E_{\mathrm{opt}}v\|_2 \leq \|Ev\|_2$ et d'après la définition de la norme de Frobenius (12.1), on a $\|E_{\mathrm{opt}}\|_F \leq \|E\|_F$ pour tout $E \in \mathcal{A}(\mathbb{S})$. Un calcul simple donne les expressions de $\|E_{\mathrm{opt}}\|_F$. ∎

Soit E_{opt} la solution optimale (solution définie par $\|E_{\mathrm{opt}}\|_F = \mu_F(y, r; \mathbb{S})$). C'est en fait la solution du problème de minimisation suivant

$$\begin{cases} \min \|E\|_F^2, \\ \text{sous } Ey - r = 0,\ E \in \mathbb{S}. \end{cases}$$

La fonctionnelle $E \mapsto \|E_{\text{opt}}\|_F^2$ est coercive et strictement convexe. De plus, l'ensemble des contraintes $\{E : Ey - r = 0, E \in \mathbb{S}\}$ est un ensemble convexe. Il s'en suit que la solution du problème existe et est unique.

Comme conséquence immédiate de (12.18) et du lemme 12.6, on obtient la caractérisation suivante de $\mu_F(y, r; \mathbb{S})$.

Théorème 12.8. *Soit $\mathbb{S}$ une algèbre de Lie ou de Jordan associée à un produit scalaire orthosymétrique $\langle \cdot, \cdot \rangle_M$ sur $\mathbf{K}^n$ avec M unitaire. Soit $A \in \mathbb{S}$ et $y \in \mathbf{K}^n$, $y \neq 0$ une solution approchée de $Ax = b$. Si y et $r := b - Ay$ satisfont les conditions du théorème 12.1 alors*

$$\mu_F(y, r; \mathbb{S}) = \frac{1}{\|y\|_2} \sqrt{2\|r\|_2^2 - \frac{|\langle y, r \rangle_M|^2}{\|y\|_2^2}},$$

et autrement $\mu_F(y, r; \mathbb{S}) = \infty$.

Comme $|\langle y, r \rangle_M| \leq \|y\|_2 \|Mr\|_2 = \|y\|_2 \|r\|_2$ et $\mu_2(y, r, \mathbb{S}) \leq \mu_F(y, r, \mathbb{S})$ on obtient la borne suivante pour l'erreur inverse structurée $\mu_\nu(y, r; \mathbb{S})$.

Théorème 12.9. *Soit $\mathbb{S}$ une algèbre de Lie ou de Jordan associée à un produit scalaire orthosymétrique $\langle \cdot, \cdot \rangle_M$ sur $\mathbf{K}^n$ avec M unitaire. Soit $A \in \mathbb{S}$ et $y \in \mathbf{K}^n$, $y \neq 0$ une solution approchée de $Ax = b$. Si y et $r := b - Ay$ satisfont les conditions du théorème 12.1, alors*

$$\mu_\nu(y, r) \leq \mu_\nu(y, r; \mathbb{S}) \leq \sqrt{2}\, \mu_\nu(y, r), \quad \nu = 2, F.$$

De plus, si $\mathbb{S}$ est telle que $M \cdot \mathbb{S} = \text{Skew}(\mathbf{K})$ alors $\mu_F(y, r; \mathbb{S}) = \sqrt{2}\, \mu_F(y, r)$.

Pour illustrer tout cela, la table 12.2 donne une expression explicite pour l'erreur inverse structurée pour un certain nombre de classes de matrices structurées. On donne aussi le ratio $\mu_F(y, r; \mathbb{S})/\mu_F(y, r)$.

12.4.2 Cas général

Quand à la fois A et b sont perturbés, on définit l'erreur inverse structurée par

$$\begin{aligned} \eta_\nu(y; \mathbb{S}) \;=\; \min\{\varepsilon :\; & (A + \Delta A)y = b + \Delta b,\ \Delta A \in \mathbb{S}, \\ & \|\Delta A\|_\nu \leq \varepsilon \|E\|_\nu,\ \|\Delta b\|_2 \leq \varepsilon \|f\|_2\}, \quad \nu = 2, F. \end{aligned} \tag{12.19}$$

Pour des perturbations non-structurées, on a [87, Thm. 7.1]

$$\eta_\nu(y) = \frac{\|r\|_2}{\|E\|_\nu \|y\|_2 + \|f\|_2}, \quad \nu = 2, F. \tag{12.20}$$

Pour $E = A$ et $f = b$ dans (12.19), $\eta_\nu(y; \mathbb{S})$ est appelée l'erreur inverse structurée relative.

Il est difficile d'obtenir une expression explicite pour $\eta_\nu(y; \mathbb{S})$. Quand $\mathbb{S}$ est une algèbre de Lie ou de Jordan d'un produit scalaire orthosymétrique, la contrainte $\Delta A \in \mathbb{S}$ impose une contrainte supplémentaire sur la perturbation Δb. En effet, d'après le théorème 12.1, un tel $\Delta A \in \mathbb{S}$ existe si et seulement si y et $r + \Delta b$ satisfont une certaine condition. Par

TAB. 12.2 – Erreur inverse structurée pour un certain nombre de structures dérivant d'algèbres de Jordan $\mathbb{J}$ et de Lie $\mathbb{L}$.

Algèbre de Jordan $\mathbb{J}$	$\mu_F(y,r;\mathbb{J})$	$\mu = \mu_F(y,r;\mathbb{J})/\mu_F(y,r)$
Symétrique complexe	$\|y\|_2^{-1}\sqrt{2\|r\|_2^2 - \|y^T r\|^2\|y\|_2^{-2}}$	$1 \leq \mu \leq \sqrt{2}$
Pseudo-symétrique complexe	$\|y\|_2^{-1}\sqrt{2\|r\|_2^2 - \|y^T \Sigma_{p,q} r\|^2\|y\|_2^{-2}}$	$1 \leq \mu \leq \sqrt{2}$
Persymétrique complexe	$\|y\|_2^{-1}\sqrt{2\|r\|_2^2 - \|y^T R r\|^2\|y\|_2^{-2}}$	$1 \leq \mu \leq \sqrt{2}$
Anti-Hamiltonienne & J-anti-symétrique	$\sqrt{2}\|y\|_2^{-1}\|r\|_2$	$\mu = \sqrt{2}$
Hermitienne	$\|y\|_2^{-1}\sqrt{2\|r\|_2^2 - (y^* r)^2\|y\|_2^{-2}}$	$1 \leq \mu \leq \sqrt{2}$
Pseudo-Hermitienne	$\|y\|_2^{-1}\sqrt{2\|r\|_2^2 - (y^* \Sigma_{p,q} r)^2\|y\|_2^{-2}}$	$1 \leq \mu \leq \sqrt{2}$
J-anti-Hermitienne	$\|y\|_2^{-1}\sqrt{2\|r\|_2^2 - \|y^* J r\|^2\|y\|_2^{-2}}$	$1 \leq \mu \leq \sqrt{2}$

Algèbre de Lie $\mathbb{L}$	$\mu_F(y,r;\mathbb{L})$	$\mu = \mu_F(y,r;\mathbb{L})/\mu_F(y,r)$
Anti-symétrique complexe	$\sqrt{2}\|y\|_2^{-1}\|r\|_2$	$\mu = \sqrt{2}$
Pseudo anti-symétrique complexe	$\sqrt{2}\|y\|_2^{-1}\|r\|_2$	$\mu = \sqrt{2}$
Per-anti-symétrique complexe	$\sqrt{2}\|y\|_2^{-1}\|r\|_2$	$\mu = \sqrt{2}$
Hamiltonienne & J-symétrique	$\|y\|_2^{-1}\sqrt{2\|r\|_2^2 - \|y^T J r\|^2\|y\|_2^{-2}}$	$1 \leq \mu \leq \sqrt{2}$
Anti-Hermitienne	$\|y\|_2^{-1}\sqrt{2\|r\|_2^2 - \|y^* r\|^2\|y\|_2^{-2}}$	$1 \leq \mu \leq \sqrt{2}$
Pseudo anti-Hermitienne	$\|y\|_2^{-1}\sqrt{2\|r\|_2^2 - \|y^* \Sigma_{p,q} r\|^2\|y\|_2^{-2}}$	$1 \leq \mu \leq \sqrt{2}$
J-Hermitienne	$\|y\|_2^{-1}\sqrt{2\|r\|_2^2 - \|y^* J r\|^2\|y\|_2^{-2}}$	$1 \leq \mu \leq \sqrt{2}$

Ici, $R = \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix}$, $\Sigma_{p,q} = \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} \in \mathbf{R}^{n\times n}$ et $J = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}$

exemple, si $\mathbb{S}$ est l'algèbre de Lie associée à une forme bilinéaire symétrique sur $\mathbf{K}^n$ alors Δb doit satisfaire $(r+\Delta b)^T y = 0$. Mais heureusement, il y a encore un nombre important d'algèbres de Lie et de Jordan pour lesquelles aucune condition supplémentaire sur Δb n'est requise (voir le théorème 12.1 et la table 12.1 pour des exemples de telles structures).

Dans les cas où y et r satisfont les conditions du théorème 12.1, on obtient les bornes suivantes pour l'erreur inverse structurée relative.

Théorème 12.10. *Soit $\mathbb{S}$ une algèbre de Lie ou de Jordan associée à un produit scalaire orthosymétrique $\langle\cdot,\cdot\rangle_M$ sur $\mathbf{K}^n$ avec M unitaire. Soit $A \in \mathbb{S}$ et $y \in \mathbf{K}^n$, $y \neq 0$ une solution approchée de $Ax = b$. Supposons que $E = A$ et $f = b$ dans (12.19). Alors si y et $r := b - Ay$ satisfont les conditions du théorème 12.1, on a*

$$\eta_\nu(y) \leq \eta_\nu(y;\mathbb{S}) \leq \frac{2\sqrt{2}\,\eta_\nu(y)}{1-\eta_\nu(y)}. \tag{12.21}$$

Démonstration. D'après la définition de $\eta_\nu(y;\mathbb{S})$ et de $\mu_\nu(y,r;\mathbb{S})$ et d'après le théorème 12.9, on a

$$\eta_\nu(y) \leq \eta_\nu(y;\mathbb{S}) \leq \frac{\mu_\nu(r,y;\mathbb{S})}{\|E\|_\nu} \leq \sqrt{2}\frac{\mu_\nu(r,y)}{\|E\|_\nu} = \sqrt{2}\frac{\|r\|_2}{\|E\|_\nu\|y\|_2}. \tag{12.22}$$

Soit $\varepsilon = \eta_\nu(y,r)$. Alors d'après (12.20), on a $\|r\|_2 \leq \varepsilon(\|A\|_\nu\|y\|_2 + \|b\|_2)$ avec $\|b\|_2 = \|(A+\Delta A)y - \Delta b\|_2 \leq (1+\varepsilon)\|A\|_\nu\|y\|_2 + \varepsilon\|b\|_2$ ce qui entraîne que $\|b\|_2 \leq (1-\varepsilon)^{-1}(1+\varepsilon)\|A\|_\nu\|y\|_2$. Ainsi

$$\|r\|_2 \leq \frac{2\varepsilon}{1-\varepsilon}\|A\|_\nu\|y\|_2$$

et l'equation (12.22) implique la borne d'erreur. ■

Les inégalités (12.21) montrent que quand l'erreur inverse non structurée relative $\eta_\nu(y)$ est petite, alors l'erreur inverse non structurée relative $\eta_\nu(y;\mathbb{S})$ est aussi petite. Néanmoins, il faut être prudent dans l'interprétation des résultats du théorème 12.10. La borne supérieure n'est valide que si $\mu_\nu(y,r;\mathbb{S})$ est finie, c.-à-d., si y et r satisfont les conditions d'existence du théorème 12.1. Par exemple, quand $\mathbb{S}$ est l'algèbre de Jordan associée à une forme bilinéaire symétrique ou bien l'algèbre de Lie associée à une forme bilinéaire anti-symétrique, alors les bornes sont toujours vraies. (12.21) C'est encore le cas pour les structures symétriques, symétrique complexes, pseudo-symétriques, persymétrique, Hamiltoniennes et J-symétriques complexes. Dans ces cas, résoudre un système linéaire avec un algorithme inverse-stable pour des perturbations non-structurées donne une solution approchée avec une faible erreur inverse structurée.

12.4.3 Erreur inverse structurée particulière

Dans cette sous-section, $\mathbb{S}$ est l'algèbre de Jordan associée à une forme bilinéaire symétrique sur $\mathbf{R}^n$ ou bien l'algèbre de Lie associée à une forme anti-symétrique sur $\mathbf{R}^n$, c.-à-d., $M\cdot\mathbb{S} = \mathrm{Sym}(\mathbf{R})$. Des exemples de telles matrices incluent les matrices symétriques, pseudo-symétriques, persymétrique et Hamiltoniennes.

Quand à la fois A et b sont perturbés, on définit l'erreur inverse structurée comme dans [187] par

$$\widetilde{\eta}_F(y,\theta;\mathbb{S}) = \min\{\|\Delta A, \theta\Delta b\|_F : \ (A+\Delta A)y = b+\Delta b, \ \Delta A \in \mathbb{S}\}, \quad \theta > 0. \tag{12.23}$$

Le paramètre θ permet une certaine flexibilité sur les mesures des perturbations. Par exemple, quand $\theta = \|A\|_F\|b\|_2$,

$$\frac{1}{\|A\|_F}\widetilde{\eta}(y, \|A\|_F\|b\|_2, \mathbb{S}) = \min\left\{ \left\| \left(\frac{\|\Delta A\|_F}{\|A\|_F}, \frac{\|\Delta b\|_2}{\|b\|_2} \right)^T \right\|_2 : \ (A+\Delta A)y = b+\Delta b, \ \Delta A \in \mathbb{S} \right\}$$

donne une erreur inverse pour lesquelles les deux perturbations sont mesurées de manière relative et « normwise ».

Il est plus facile d'obtenir une expression explicite pour $\widetilde{\eta}_F(y,\theta;\mathbb{S})$ que pour $\eta_\nu(y;\mathbb{S})$.

Théorème 12.11. *Soit $\mathbb{S}$ une algèbre de Lie ou de Jordan associée à un produit scalaire orthosymétrique $\langle\cdot,\cdot\rangle_M$ sur $\mathbf{K}^n$ avec M unitaire tel que $M\cdot\mathbb{S} = \mathrm{Sym}(\mathbf{R})$. Soit $A \in \mathbb{S}$ et $y \in \mathbf{K}^n$, $y \neq 0$ une solution approchée de $Ax = b$. Alors*

$$\widetilde{\eta}_F(y,\theta;\mathbb{S}) = \frac{\theta}{\sqrt{1+\theta^2\|y\|_2^2}} \sqrt{\|r\|_2^2 + \frac{\theta^2[(\|y\|_2\|r\|_2)^2 - (y^TMr)^2]}{2+\theta^2\|y\|_2^2}}$$

et

$$\widetilde{\eta}_F(y,\theta) \leq \widetilde{\eta}_F(y,\theta;\mathbb{S}) \leq \sqrt{2}\,\widetilde{\eta}_F(y,\theta), \quad \forall\ \theta > 0.$$

Démonstration. Soit $\mathcal{A}(\mathbb{S}) := \{E \in \mathbb{S} : Ey = r + \Delta b\}$. On a

$$\begin{aligned}
(\widetilde{\eta}_F(y,\theta;\mathbb{S}))^2 &= \min\{\|\Delta A, \theta\Delta b\|_F^2 : \ \Delta A y = r+\Delta b, \ \Delta A \in \mathbb{S}\} \\
&= \min_{\Delta b\in\mathbf{R}^n} \min_{\Delta A\in\mathcal{A}(\mathbb{S})} \{\|\Delta A\|_F^2 + \theta^2\|\Delta b\|_2^2\} \\
&= \min_{\Delta b\in\mathbf{R}^n} \left\{ \theta^2\|\Delta b\|_2^2 + \min_{\Delta A\in\mathcal{A}(\mathbb{S})} \|\Delta A\|_F^2 \right\} \\
&= \min_{\Delta b\in\mathbf{R}^n} \left\{ \theta^2\|\Delta b\|_2^2 + [\mu_F(y, r+\Delta b;\mathbb{S})]^2 \right\} := \min_{\Delta b\in\mathbf{R}^n} f(\Delta b;\mathbb{S}).
\end{aligned}$$

De plus, $\mu_F(y, r+\Delta b;\mathbb{S}) = \mu_F(y, \widetilde{r} + \widetilde{\Delta b}; \mathrm{Sym}(\mathbf{R}))$ avec $\widetilde{r} = Mr$ et $\widetilde{\Delta b} = M\Delta b$. Ainsi,

$$f(\Delta b;\mathbb{S}) = \theta^2\|\widetilde{\Delta b}\|_2^2 + \mu_F(y, \widetilde{r} + \widetilde{\Delta b}; \mathrm{Sym}(\mathbf{R})) = f(\widetilde{\Delta b}; \mathrm{Sym}(\mathbf{R})).$$

Sun [187] a montré que le minimum de $f(\widetilde{\Delta b}; \mathrm{Sym}(\mathbf{R}))$ sur $\widetilde{\Delta b} \in \mathbf{R}^n$ est atteint par

$$\widetilde{\Delta b}_{\min} = \frac{\theta^2(y^T\widetilde{r})y - 2(1+\theta^2\|y\|_2^2)\widetilde{r}}{(2+\theta^2\|y\|_2^2)(1+\theta^2\|y\|_2^2)}$$

si bien que

$$\begin{aligned}
(\widetilde{\eta}_F(y,\theta;\mathbb{S}))^2 &= \frac{\theta^2\Big(2(1+\theta^2\|y\|_2^2)\|\widetilde{r}\|_2^2 - \theta^2(y^T\widetilde{r})^2\Big)}{(2+\theta^2\|y\|_2^2)(1+\theta^2\|y\|_2^2)} \\
&= \frac{\theta^2\Big(2(1+\theta^2\|y\|_2^2)\|r\|_2^2 - (\theta/b)^2(y^TMr)^2\Big)}{(2+\theta^2\|y\|_2^2)(1+\theta^2\|y\|_2^2)}.
\end{aligned}$$

Un majorant pour $\widetilde{\eta}_F(y,\theta;\mathbb{S})$ est obtenu en utilisant l'inégalité

$$0 \leq \frac{\theta^2[(\|y\|_2\|r\|_2)^2 - (y^T M r)^2]}{2+\theta^2\|y\|_2^2} \leq \|r\|_2^2.$$

■

12.5 Remarques et conclusion

Dans ce chapitre, nous avons montré que pour des structures dérivant d'algèbres de Lie ou de Jordan associées à un produit scalaire orthosymétrique, il y a peu ou pas de différences entre le conditionnement structuré et non-structuré pour la résolution de systèmes linéaires et l'inversion de matrices ainsi que pour la distance à la singularité. De plus, nous avons obtenu des bornes sur le ratio entre le conditionnement structuré et non-structuré. Grâce à la théorie des variétés lisses, nous avons obtenu une expression calculable pour le conditionnement structuré des systèmes linéaires et de l'inversion pour des structures non-linéaires avec la norme de Frobenius.

Nous avons aussi considéré l'erreur inverse structurée à la fois pour les systèmes linéaires et les problèmes de valeurs propres. Nous avons obtenu des formules explicites et des ratios avec le cas non-structuré.

Nous espérons avoir montré que les algèbres de Lie et de Jordan sont un bon cadre pour traiter de certaines matrices structurées et ainsi de généraliser les théorèmes existants à de nouvelles structures.

Un autre problème intéressant d'optimisation est celui de trouver la matrice structurée la plus proche d'un certaine matrice non-structurée A

$$d_{\mathbb{S}}(A) := \min\{\|A-S\| : \ S \in \mathbb{S}\}.$$

Ce problème est facile à résoudre quand $\mathbb{S}$ est une algèbre de Lie ou de Jordan associée à un produit scalaire orthosymétrique $\langle\cdot,\cdot\rangle_{\text{M}}$ sur $\mathbf{K}^n$ avec M unitaire et $\|\cdot\|$ est une norme unitairement invariante. En effet, si $\langle\cdot,\cdot\rangle_{\text{M}}$ est orthosymétrique alors tout matrice $A \in \mathbf{K}^{n\times n}$ peut s'écrire sous la forme

$$A = \frac{A+A^\star}{2} + \frac{A-A^\star}{2} =: A_{\mathbb{J}} + A_{\mathbb{L}}$$

avec $A_{\mathbb{J}} \in \mathbb{J}$ et $A_{\mathbb{L}} \in \mathbb{L}$. Alors si $\mathbb{S} = \mathbb{J}$ pour tout $S \in \mathbb{J}$ $(S = S^\star)$, on a

$$\begin{aligned} \|A - A_{\mathbb{J}}\| = \|A_{\mathbb{L}}\| &= \frac{1}{2}\|A - A^\star\| = \frac{1}{2}\|(A-S) + (S^\star - A^\star)\| \\ &\leq \frac{1}{2}\|A-S\| + \frac{1}{2}\|(S-A)^\star)\| \leq \|A-S\| \end{aligned}$$

puisque M is unitaire et la norme unitairement invariante. Par conséquent, pour les algèbres de Jordan, $d_{\mathbb{J}}(A) = \frac{1}{2}\|A-A^\star\|$ et de façon similaire pour les algèbres de Lie, $d_{\mathbb{L}}(A) = \frac{1}{2}\|A+A^\star\|$. Fan et Hoffman ont résolu ce problème pour les matrices symétriques [59] et plus récemment Cardoso, Kenney et Leite [35, Thm. 5.3] l'ont resolu pour les algèbres de Lie et de Jordan avec un produit scalaire $\langle\cdot,\cdot\rangle_{\text{M}}$ pour lesquel $MM^T = I$ et $M^2 = \pm I$. Il s'agit de conditions plus fortes que les notre.

Conclusion et perspectives

Cette thèse s'est articulée autour de méthodes et d'outils pour étudier la précision et la stabilité d'algorithmes numériques. Nous avons classé nos contributions en trois grandes catégories que nous allons développer maintenant.

1 Contributions

Évaluation polynomiale

Dans cette thèse, nous nous sommes particulièrement intéressés à la notion d'analyse d'erreur structurée. Il s'agit d'un domaine récent et porteur. Notre contribution en algèbre polynomiale numérique a été de considérer les perturbations réelles de polynômes réels car auparavant seules des perturbations complexes indépendantes du type (réels ou complexes) des coefficients des polynômes étaient étudiées. Notre analyse a montré que pour le conditionnement, il y avait en général très peu de différence mais que cette différence pouvait devenir importante pour l'erreur inverse.

La précision du résultat d'un calcul en précision finie dépend de trois facteurs : le conditionnement du problème à résoudre, la stabilité de l'algorithme numérique et la précision de l'arithmétique utilisée. Cette dernière est souvent l'arithmétique flottante binaire IEEE 754 qui permet une précision maximale de l'ordre de 16 chiffres décimaux en format long (double précision). Cette précision n'est pas suffisante si on cherche une solution précise de problèmes mal conditionnés. Un certain nombre de solutions ont été proposées pour permettre d'augmenter la précision de l'arithmétique utilisée comme par exemple les « double-double ». Notre objectif a été de proposer une alternative aux « double-double » qui soit à la fois performante, fiable et portable. Nous nous sommes plus particulièrement intéressés à l'évaluation polynomiale par la méthode de Horner. Nous avons proposé un algorithme rapide et précis ; rapide au sens ou notre algorithme est deux fois plus rapide qu'avec les « double-double », précis au sens ou le résultat est aussi précis que s'il avait été calculé avec une précision double de la précision courante. Nous avons aussi proposé une borne validée sur l'erreur commise durant l'évaluation.

En utilisant les mêmes techniques, nous avons proposé un algorithme rapide et robuste

pour le calcul du prédicat géométrique ORIENT3D. Bien entendu, les mêmes techniques s'appliquent aisément pour les autres prédicats comme INCIRCLE et INSPHERE. Le principal potentiel de notre algorithme est de fournir une méthode simple et efficace pour augmenter la précision des variables critiques dans les algorithmes en géométrie algorithmique. Les techniques que nous avons utilisées sont suffisamment simples pour être implémentées facilement sur n'importe quelle architecture.

Pseudozéros et applications

Contrairement à la notion de pseudospectre, les pseudozéros ont été très peu étudiés. Or nous nous sommes rendus compte qu'ils pouvaient être utiles en théorie du contrôle et en calcul formel. En théorie du contrôle, nous avons donné un algorithme permettant de tester la stabilité d'un polynôme. Nous avons aussi programmé notre outil PSIP pour tracer les pseudozéros d'un polynôme d'intervalle, ce qui a aussi des applications en contrôle [107]. En calcul formel, nous avons décrit un algorithme permettant de tester la primalité approchée de deux polynômes.

Pseudospectres et conditionnement matriciel

Les racines d'un polynôme sont les valeurs propres de sa matrice compagnon. On peut donc ramener un problème polynomial à un problème matriciel. La généralisation des pseudozéros pour les matrices est le pseudospectre.

Nous avons introduit la notion de pseudospectre structuré, c'est à dire que nous autorisons seulement des perturbations conservant la structure de la matrice. Nous avons montré que le pseudospectre et le pseudospectre structuré coïncidaient pour les structures classique de type Toeplitz, Hankel, circulante.

Pour ce qui est des systèmes linéaires, nous nous sommes intéressés aux perturbations de matrices provenant d'algèbres de Lie et de Jordan. Ce formalisme, assez récent dans son utilisation comme structure de perturbation, permet de généraliser des travaux plus anciens à beaucoup de structures différentes. Ce cadre nous a permis en particulier de généraliser des travaux de Rump [172] et Higham [82, 83].

2 Perspectives

Nous présentons succinctement ici les grands axes de nos travaux futurs.

Analyse d'erreur structurée

Les perspectives de ce travail sont multiples. D'un point de vue de l'analyse d'erreur structurée, nous nous sommes intéressé à la résolution de systèmes linéaires, au calcul de l'inverse d'une matrice et à la recherche de valeurs propres. Il reste encore beaucoup d'autres algorithmes à considérer comme par exemple le problème de moindres carrés et donc naturellement les problèmes du calcul de l'inverse généralisé (aussi dit pseudo-inverse) de Penrose-Moore.

Pseudozéros certifiés et pseudozéros de polynômes généralisés

À l'heure actuelle, le tracé des pseudozéros n'est pas certifié. Il s'agit d'une lacune importante. Néanmoins, l'analyse par intervalles et des algorithmes comme Sivia devraient permettre de pouvoir certifier les tracés.

Dans cette thèse, nous nous sommes intéressés aux pseudozéros de polynômes. Or dans bon nombre d'applications, on voit apparaître plutôt ce qu'on appelle des *polynômes généralisés* [60]. Dans ce cas, les monômes ne sont plus de la forme z^i mais de la forme e^{iz} ou bien $\cos(iz)$. On peut généraliser la notion de pseudozéros à des tels polynômes généralisés.

Amélioration de la précision

À l'heure actuelle est en train d'émerger un nouveau domaine, le GPGPU : General-Purpose Computation Using Graphics Hardware. L'idée est d'utiliser la puissance des unités flottantes des cartes graphiques actuelles qui sont souvent inexploitées. Le problème est que ces unités flottantes utilisent la représentation de la norme IEEE 754 pour la simple précision (elles n'adhèrent cependant pas au standard IEEE 754 pour les opérations flottantes). Or pour les applications actuelles, la double précision s'impose. Nous pensons pouvoir appliquer les méthodes d'augmentation de la précision pour pouvoir émuler la double précision sur les cartes graphiques.

La première chose à faire serait de faire du raffinement itératif pour résoudre les systèmes linéaires sur GPU. En utilisant l'algorithme de sommation et de calcul de produit scalaire d'Ogita, Rump et Oishi [153], on peut calculer le résidu en précision doublée en utilisant seulement la simple précision. Il est encore bien connu [87] que l'algorithme converge vers un résultat qui est aussi précis que s'il avait été calculé avec une précision doublée.

On pourra bien sûr généraliser cela à la méthode de Newton [190] en calculant le résidu en précision doublée. Cela permettra de faire du raffinement itératif pour les problèmes de valeurs propres et de valeurs propres généralisées [87, 190].

Certification numérique

Soit un système linéaire $Ax = b$, avec $A \in \mathbf{R}^{n\times n}$ et $b \in \mathbf{R}^n$. La résolution numérique en précision finie de ce système ne donne en général pas x mais une valeur approchée $\widetilde{x}$. De nombreux articles [150, 156, 151, 152, 154] traitent du calcul d'une borne certifié en précision finie pour $\|x - \widetilde{x}\|_\infty$. Il nous semble intéressant d'obtenir des résultats similaires pour les racines de polynômes. Étant donné un polynôme p à coefficients réels ou complexes avec pour racines $(z_1, \ldots, z_n)$ et $(\widetilde{z}_1, \ldots, \widetilde{z}_n)$, n racines approchées de p, on aimerait pouvoir donner une borne certifiée en flottant qui majore $\max_{i=1:n} |\widetilde{z}_i - z_i|$. Des travaux dans ce sens ont été fait sur les valeurs propres [155] mais il semble difficile de les appliquer à la matrice compagnon et de revenir ensuite dans le cadre polynomial.

Bibliographie

[1] *GMP, the GNU multi-precision library* – Disponible à l'URL = http://www.swox.com/gmp/.

[2] *The MPFR library* – Disponible à l'URL = http://www.mpfr.org.

[3] O. Aberth – *Precise numerical methods using C++*, Academic Press Inc., San Diego, CA, 1998.

[4] I. J. Anderson – « A distillation algorithm for floating-point summation », *SIAM J. Sci. Comput.* **20** (1999), no. 5, p. 1797–1806 (electronic).

[5] E. Artin – *Algèbre géométrique*, french éd., Les Grands Classiques Gauthier-Villars, Éditions Jacques Gabay, Paris, 1996, Translated from the 1957 English original by M. Lazard, Edited and with a foreword by G. Julia.

[6] F. Avnaim, J.-D. Boissonnat, O. Devillers, F. P. Preparata et M. Yvinec – « Evaluating signs of determinants using single-precision arithmetic », *Algorithmica* **17** (1997), no. 2, p. 111–132.

[7] D. H. Bailey – « Algorithm 719 ; multiprecision translation and execution of Fortran programs », *ACM Trans. Math. Softw.* **19** (1993), no. 3, p. 288–319.

[8] — , « A Fortran 90-based multiprecision system », *ACM Trans. Math. Softw.* **21** (1995), no. 4, p. 379–387.

[9] — , *A Fortran-90 double-double library*, 2001, Disponible à l'URL = http://crd.lbl.gov/~dhbailey/mpdist/index.html.

[10] R. Barrio – « A unified rounding error bound for polynomial evaluation », *Adv. Comput. Math.* **19** (2003), no. 4, p. 385–399.

[11] B. Beckermann et G. Labahn – « A fast and numerically stable Euclidean-like algorithm for detecting relatively prime numerical polynomials », *J. Symbolic Comput.* **26** (1998), no. 6, p. 691–714.

[12] — , « When are two numerical polynomials relatively prime ? », *J. Symbolic Comput.* **26** (1998), no. 6, p. 677–689.

[13] M. Berger et B. Gostiaux – *Géométrie différentielle : variétés, courbes et surfaces*, second éd., Mathématiques, Presses Universitaires de France, Paris, 1992.

[14] D. A. Bini et G. Fiorentino – « Design, analysis, and implementation of a multiprecision polynomial rootfinder », *Numer. Algorithms* **23** (2000), no. 2-3, p. 127–173.

[15] S. Boldo et J.-M. Muller – « Some functions computable with a fused-mac », *Proceedings of the 17th Symposium on Computer Arithmetic* (Cape Cod, USA), 2005.

[16] F. Bornemann, D. Laurie, S. Wagon et J. Waldvogel – *The SIAM 100-digit challenge*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2004.

[17] T. Boros, T. Kailath et V. Olshevsky – « Pivoting and backward stability of fast algorithms for solving Cauchy linear equations », *Linear Algebra Appl.* **343/344** (2002), p. 63–99, Special issue on structured and infinite systems of linear equations.

[18] A. Böttcher, M. Embree et V. I. Sokolov – « On large Toeplitz band matrices with an uncertain block », *Linear Algebra Appl.* **366** (2003), p. 87–97, Special issue on structured matrices : analysis, algorithms and applications (Cortona, 2000).

[19] A. Böttcher et S. Grudsky – *Spectral properties of banded Toeplitz matrices*, 2005, Book to appear.

[20] A. Böttcher, S. Grudsky et A. Kozak – « On the distance of a large Toeplitz band matrix to the nearest singular matrix », Toeplitz matrices and singular integral equations (Pobershau, 2001), Oper. Theory Adv. Appl., vol. 135, Birkhäuser, Basel, 2002, p. 101–106.

[21] S. Boyd et V. Balakrishnan – « A regularity result for the singular values of a transfer matrix and a quadratically convergent algorithm for computing its L_∞-norm », *Syst. Control Lett.* **15** (1990), no. 1, p. 1–7.

[22] S. Boyd, V. Balakrishnan et P. Kabamba – « A bisection method for computing the H_∞ norm of a transfer matrix and related problems », *Math. Control Signals Systems* **2** (1989), no. 3, p. 207–219.

[23] R. P. Brent – « A fortran multiple-precision arithmetic package », *ACM Trans. Math. Softw.* **4** (1978), no. 1, p. 57–70.

[24] K. Briggs – *Doubledouble floating point arithmetic*, 1998, Disponible à l'URL = http://keithbriggs.info/.

[25] H. Brönnimann et M. Yvinec – « Efficient exact evaluation of signs of determinants », *SCG '97 : Proceedings of the thirteenth annual symposium on Computational geometry* (New York, NY, USA), ACM Press, 1997, p. 166–173.

[26] — , « Efficient exact evaluation of signs of determinants », *Algorithmica* **27** (2000), no. 1, p. 21–56.

[27] J. R. Bunch, J. W. Demmel et C. F. Van Loan – « The strong stability of algorithms for solving symmetric linear systems », *SIAM J. Matrix Anal. Appl.* **10** (1989), no. 4, p. 494–499.

[28] J. V. Burke, A. S. Lewis et M. L. Overton – « Optimization and pseudospectra, with applications to robust stability », *SIAM J. Matrix Anal. Appl.* **25** (2003), no. 1, p. 80–104.

[29] — , « Robust stability and a criss-cross algorithm for pseudospectra », *IMA J. Numer. Anal.* **23** (2003), no. 3, p. 359–375.

[30] — , « Variational analysis of the abscissa mapping for polynomials via de Gauss-Lucas theorem », *J. Global Optim.* **28** (2004), no. 3-4, p. 259–268.

[31] J. V. Burke et M. L. Overton – « Variational analysis of the abscissa mapping for polynomials », *SIAM J. Control Optimization* **39** (2001), no. 6, p. 1651–1676.

[32] R. Byers – « A bisection method for measuring the distance of a stable matrix to the unstable matrices », *SIAM J. Sci. Stat. Comput.* **9** (1988), no. 5, p. 875–881 (English).

[33] R. Byers et D. Kressner – « On the condition of a complex eigenvalue under real perturbations », *BIT* **44** (2004), no. 2, p. 209–214.

[34] S. Cabay, A. R. Jones et G. Labahn – « Algorithm 766 : Experiments with a weakly stable algorithm for computing Pade-Hermite and simultaneous Pade approximants », *ACM Trans. Math. Softw.* **23** (1997), no. 1, p. 91–110.

[35] J. R. Cardoso, C. S. Kenney et F. S. Leite – « Computing the square root and logarithm of a real P-orthogonal matrix », *Appl. Numer. Math.* **46** (2003), no. 2, p. 173–196.

[36] F. Chaitin-Chatelin et V. Frayssé – *Lectures on finite precision computations*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996.

[37] P. Chin, R. M. Corless et G. F. Corliss – « Optimization strategies for the approximate GCD problem », *Proceedings of the 1998 international symposium on symbolic and algebraic computation, ISSAC '98* (O. Gloor, éd.), Août 1998, p. 228–235.

[38] K. L. Clarkson – « Safe and effective determinant evaluation. », 33rd annual symposium on Foundations of computer science (FOCS). Proceedings, Pittsburgh, PA, USA, October 24–27, 1992. Washington, DC : IEEE Computer Society Press, 387-395 .

[39] R. M. Corless, P. M. Gianni, B. M. Trager et S. M. Watt – « The singular value decomposition for polynomial systems », *Proceedings of the 1995 international symposium on symbolic and algebraic computation, ISSAC '95* (A. H. M. Levelt, éd.), ACM Press, Juillet 1995, p. 195–207.

[40] R. M. Corless, H. Kai et S. M. Watt – « Approximate computation of pseudovarieties », *SIGSAM Bull.* **37** (2003), no. 3, p. 67–71.

[41] M. Daumas, F. de Dinechin et A. Tisserand – « L'arithmétique des ordinateurs », *Réseaux et Systèmes Répartis, Calculateurs Parallèles* **13** (2001), no. 4-5, p. 327–552.

[42] M. Daumas et C. Finot – « Division of floating point expansions with an application to the computation of a determinant. », *J. UCS* **5** (1999), no. 6, p. 323–338 (English).

[43] J.-P. Dedieu, M.-H. Kim, M. Shub et F. Tisseur – « Implicit gamma theorems. I. Pseudoroots and pseudospectra », *Found. Comput. Math.* **3** (2003), no. 1, p. 1–31.

[44] D. Defour – « Collapsing dependent floating point operations », *IMACS World Congress Scientific Computation, Applied Mathematics and Simulation* (Paris, France), July 2005.

[45] D. Defour et B. Goossens – « Implémentation de l'opérateur add2 », Rapport de Recherche 03, Équipe de recherche DALI, Laboratoire LP2A, Université de Perpignan Via Domitia, France, 52 avenue Paul Alduy, 66860 Perpignan cedex, France, december 2004.

[46] T. J. Dekker – « A floating-point technique for extending the available precision », *Numer. Math.* **18** (1971), p. 224–242.

[47] J. W. Demmel – « Underflow and the reliability of numerical software », *SIAM J. Sci. Statist. Comput.* **5** (1984), no. 4, p. 887–919.

[48] — , « On condition numbers and the distance to the nearest ill-posed problem », *Numer. Math.* **51** (1987), no. 3, p. 251–289.

[49] — , *Applied numerical linear algebra*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[50] — , « Accurate singular value decompositions of structured matrices », *SIAM J. Matrix Anal. Appl.* **21** (1999), no. 2, p. 562–580 (electronic).

[51] J. W. Demmel et Y. Hida – « Accurate and efficient floating point summation », *SIAM J. Sci. Comput.* **25** (2003), no. 4, p. 1214–1248 (electronic).

[52] — , « Fast and accurate floating point summation with application to computational geometry », *Numer. Algorithms* **37** (2004), no. 1-4, p. 101–112.

[53] A. Dickenstein et I. Z. Emiris (éds.) – *Solving polynomial equations : Foundations, algorithms, and applications*, Algorithms and Computation in Mathematics, vol. 14, Springer-Verlag, Berlin, 2005.

[54] A. Edelman et H. Murakami – « Polynomial roots from companion matrix eigenvalues », *Math. Comp.* **64** (1995), no. 210, p. 763–776.

[55] B. Einarsson, R. Boisvert, F. Chaitin-Chatelin, R. Cools, C. Douglas, K. Dritz, W. Enright, W. Gropp, S. Hammarling, H. P. Langtangen, R. Pozo, S. Rump, V. Snyder, E. Traviesas-Cassan, M. Vouk, W. Walster et B. Wichmann – *Accuracy and reliability in scientific computing*, Software-Environments-Tools, SIAM, Philadelphia, PA, 2005.

[56] I. Z. Emiris – « Symbolic-numeric algebra for polynomials », Rapport de recherche, Institut National de Recherche en Informatique et en Automatique (INRIA), 1997.

[57] I. Z. Emiris, A. Galligo et H. Lombardi – « Numerical univariate polynomial GCD », The mathematics of numerical analysis (Park City, UT, 1995), Amer. Math. Soc., Providence, RI, 1996, p. 323–343.

[58] — , « Certified approximate univariate GCDs », *J. Pure Appl. Algebra* **117/118** (1997), p. 229–251.

[59] K. Fan et A. J. Hoffman – « Some metric inequalities in the space of matrices », *Proc. Amer. Math. Soc.* **6** (1955), p. 111–116.

[60] A. Frommer – « A unified approach to methods for the simultaneous computation of all zeros of generalized polynomials », *Numer. Math.* **54** (1988), no. 1, p. 105–116.

[61] W. Gautschi – « On the condition of algebraic equations », *Numer. Math.* **21** (1973), p. 405–424.

[62] — , « Questions of numerical condition related to polynomials », Studies in numerical analysis, Math. Assoc. America, Washington, DC, 1984, p. 140–177.

[63] K. O. Geddes, S. R. Czapor et G. Labahn – *Algorithms for computer algebra*, Kluwer Academic Publishers, Boston, MA, 1992.

[64] Y. Genin, R. Ştefan et P. Van Dooren – « Real and complex stability radii of polynomial matrices », *Linear Algebra Appl.* **351/352** (2002), p. 381–410, Fourth special issue on linear systems and control.

[65] D. Goldberg – « What every computer scientist should know about floating-point arithmetic », *ACM Computing Surveys* **23** (1991), no. 1, p. 5–48.

[66] G. H. Golub et C. F. Van Loan – *Matrix computations*, third éd., Johns Hopkins University Press, Baltimore, MD, 1996.

[67] L. Gonzalez-Vega et I. Necula – « Efficient topology determination of implicitly defined algebraic plane curves », *Comput. Aided Geom. Design* **19** (2002), no. 9, p. 719–743.

[68] J. Grabmeier, E. Kaltofen et V. Weispfenning (éds.) – *Computer algebra handbook*, Springer-Verlag, Berlin, 2003, Foundations. Applications. Systems, With 1 CD-ROM (Windows, Macintosh).

[69] S. Graillat – « A note on structured pseudospectra », *J. Comput. Appl. Math.* (2005), à paraître.

[70] S. Graillat et P. Langlois – « Testing polynomial primality with pseudozeros », *Proceedings of the Fifth Conference on Real Numbers and Computers* (Lyon, France), September 2003, p. 231–246.

[71] — , « Computation of stability radius for polynomials », Prépublication No31, Laboratoire MANO, Janvier 2004.

[72] — , « More on pseudozeros for univariate polynomials », Prépublication No32, Laboratoire MANO, Janvier 2004, Soumis à Theoretical Computer Science.

[73] — , « Pseudozero set of interval polynomials », *Proceedings of the 21th ACM Symposium on Applied Computing SAC'2006, Dijon, France*, Avril 2006, À paraître.

[74] S. Graillat et N. Louvet – « Applications of fast and accurate summation in computational geometry », Rapport de recherche, Équipe de recherche DALI, Laboratoire LP2A, Université de Perpignan Via Domitia, France, 52 avenue Paul Alduy, 66860 Perpignan cedex, France, September 2005, soumis à SAC 2005.

[75] S. Graillat, N. Louvet et P. Langlois – « Compensated Horner scheme », Rapport de recherche 04, Équipe de recherche DALI, Laboratoire LP2A, Université de Perpignan Via Domitia, France, 52 avenue Paul Alduy, 66860 Perpignan cedex, France, Juillet 2005, Soumis à SIAM Journal of Scientific Computing.

[76] M. Grimmer, K. Petras et N. Revol – « Multiple precision interval packages : Comparing different approaches », *Lecture Notes in Computer Science* **2991** (2004), p. 64–90.

[77] M. Gu – « Stable and efficient algorithms for structured systems of linear equations », *SIAM J. Matrix Anal. Appl.* **19** (1998), no. 2, p. 279–306 (electronic).

[78] G. I. Hargreaves – « Interval analysis in MATLAB », Numerical Analysis Report No. 416, Manchester Centre for Computational Mathematics, Manchester, England, december 2002, http://www.maths.man.ac.uk/~nareports/narep416.pdf.

[79] J. R. Hauser – « Handling floating-point exceptions in numeric programs », *ACM Trans. Program. Lang. Syst.* **18** (1996), no. 2, p. 139–174.

[80] C. He et G. Watson – « An algorithm for computing the distance to instability », *SIAM J. Matrix Anal. Appl.* **20** (1998), no. 1, p. 101–116 (English).

[81] Y. Hida, X. S. Li et D. H. Bailey – « Algorithms for quad-double precision floating point arithmetic », *Proc. 15th IEEE Symposium on Computer Arithmetic*, IEEE Computer Society Press, Los Alamitos, CA, USA, 2001, p. 155–162.

[82] D. J. Higham – « Condition numbers and their condition numbers », *Linear Algebra Appl.* **214** (1995), p. 193–213.

[83] D. J. Higham et N. J. Higham – « Backward error and condition of structured linear systems », *SIAM J. Matrix Anal. Appl.* **13** (1992), no. 1, p. 162–175.

[84] — , *MATLAB guide*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000.

[85] N. J. Higham – « Matrix nearness problems and applications », *Applications of Matrix Theory* (M. J. C. Gover et S. Barnett, éds.), Oxford University Press, 1989, p. 1–27.

[86] — , *Accuracy and stability of numerical algorithms*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996.

[87] — , *Accuracy and stability of numerical algorithms*, second éd., Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2002.

[88] N. J. Higham et F. Tisseur – « More on pseudospectra for polynomial eigenvalue problems and applications in control theory », *Linear Algebra Appl.* **351/352** (2002), p. 435–453.

[89] D. Hinrichsen et B. Kelb – « Spectral value sets : a graphical tool for robustness analysis », *Systems Control Lett.* **21** (1993), no. 2, p. 127–136.

[90] D. Hinrichsen et A. J. Pritchard – « Stability radii of linear systems », *Systems Control Lett.* **7** (1986), no. 1, p. 1–10.

[91] — , « Stability radius for structured perturbations and the algebraic Riccati equation », *Systems Control Lett.* **8** (1986), no. 2, p. 105–113.

[92] — , « Real and complex stability radii : a survey », Control of uncertain systems (Bremen, 1989), Progr. Systems Control Theory, vol. 6, Birkhäuser Boston, Boston, MA, 1990, p. 119–162.

[93] — , « Robustness measures for linear systems with application to stability radii of Hurwitz and Schur polynomials », *Internat. J. Control* **55** (1992), no. 4, p. 809–844.

[94] D. Hinrichsen, B. Kelb et A. Linnemann – « An algorithm for the computation of the structured complex stability radius », *Automatica J. IFAC* **25** (1989), no. 5, p. 771–775.

[95] M. A. Hitz – « Efficient algorithms for computing the nearest polynomial with constrained roots », Thèse, Rensselaer Polytechnic Institute, Avril 1998.

[96] M. A. Hitz et E. Kaltofen – « Efficient algorithms for computing the nearest polynomial with constrained roots », *Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation (Rostock)* (New York), ACM, 1998, p. 236–243 (electronic).

[97] M. A. Hitz, E. Kaltofen et Y. N. Lakshman – « Efficient algorithms for computing the nearest polynomial with a real root and related problems », *Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation (Vancouver, BC)* (New York), ACM, 1999, p. 205–212 (electronic).

[98] J. W. Hoffman, J. J. Madden et H. Zhang – « Pseudozeros of multivariate polynomials », *Math. Comp.* **72** (2003), no. 242, p. 975–1002 (electronic).

[99] R. A. Horn et C. R. Johnson – *Matrix analysis*, Cambridge University Press, Cambridge, 1990.

[100] D. Hough – « Explaining and ameliorating the ill condition of zeros of polynomials », Thèse, University of California, Berkeley, 1977.

[101] V. Hribernig et H. J. Stetter – « Detection and validation of clusters of polynomial zeros », *J. Symbolic Comput.* **24** (1997), no. 6, p. 667–681.

[102] *WWW resources about interval arithmetic* – URL = http://www.cs.utep.edu/interval-comp/main.html.

[103] *IEEE standard for binary floating-point arithmetic, ANSI/IEEE standard 754-1985* – Institute of Electrical and Electronics Engineers, New York, 1985, Reprinted in SIGPLAN Notices, 22(2) :9–25, 1987.

[104] *A radix-independent standard for floating-point arithmetic, IEEE standard 854-1987* – IEEE Computer Society, New York, 1987.

[105] M. Jankowski, A. Smoktunowicz et H. Woźniakowski – « A note on floating-point summation of very many terms », *J. Information Processing and Cybernetics-EIK* **19** (1983), no. 9, p. 435–440.

[106] M. Jankowski et H. Woźniakowski – « The accurate solution of certain continuous problems using only single precision arithmetic », *BIT* **25** (1985), p. 635–651.

[107] L. Jaulin, M. Kieffer, O. Didrit et É. Walter – *Applied interval analysis*, Springer-Verlag London Ltd., London, 2001.

[108] C.-P. Jeannerod et G. Labahn. – « The SNAP package for arithmetic with numeric polynomials. », *Proceedings of the 2002 International Congress of Mathematical Software (ICMS'02)* (Beijing, China), August 2002.

[109] W. Kahan – « Further remarks on reducing truncation errors », *J. Assoc. Comput. Mach.* **8** (1965), no. 1, p. 40.

[110] — , « Conserving confluence curbs ill-condition », Tech. report, Computer Science Dept. Report, University of Berkeley, 1972.

[111] — , « A survey of error analysis », *Proc. IFIP Congress, Ljubljana* (Amsterdam, The Netherlands), Information Processing 71, North-Holland, 1972, p. 1214–1239.

[112] N. K. Karmarkar et Y. N. Lakshman – « Approximate polynomial greatest common divisors and nearest singular polynomials », *Proceedings of the 1996 international symposium on symbolic and algebraic computation, ISSAC '96* (Y. N. Lakshman, éd.), Juillet 1996, p. 35–39.

[113] — , « On approximate GCDs of univariate polynomials », *J. Symbolic Comput.* **26** (1998), no. 6, p. 653–666, Symbolic numeric algebra for polynomials.

[114] M. Karow – « Geometry of spectral value sets », Thèse, Universität Bremen, 2003.

[115] M. Karow, D. Kressner et F. Tisseur – « Structured eigenvalue condition numbers », Numerical Analysis Report No. 467, Manchester Centre for Computational Mathematics, Manchester, England, April 2005.

[116] D. E. Knuth – *The art of computer programming, volume 2, seminumerical algorithms*, third éd., Addison-Wesley, Reading, MA, USA, 1998.

[117] S. Krishnan, M. Foskey, T. Culver, J. Keyser et D. Manocha – « Precise : efficient multiprecision evaluation of algebraic roots and predicates for reliable geometric computation », *SCG '01 : Proceedings of the seventeenth annual symposium on Computational geometry* (New York, NY, USA), ACM Press, 2001, p. 274–283.

[118] J. Lafontaine – *Introduction aux variétés différentielles*, Grenoble Sciences, EDP Sciences, 1996.

[119] P. Lancaster et P. Psarrakos – « On the pseudospectra of matrix polynomials », Numerical Analysis Report No. 445, Manchester Centre for Computational Mathematics, Manchester, England, february 2004, http://www.maths.man.ac.uk/~nareports/narep445.pdf.

[120] P. Langlois – « Automatic linear correction of rounding errors », *BIT* **41** (2001), no. 3, p. 515–539.

[121] — , « Précision finie et méthodes automatiques », Habilitation à diriger des recherches, Université Claude Bernard Lyon 1, Juillet 2001.

[122] — , « Analyse d'erreur en précision finie », Outils d'analyse numérique pour l'Automatique (A. Barraud, éd.), Traité IC2, Hermès Science, 2002, p. 19–52.

[123] — , « More accuracy at fixed precision », *J. Comput. Appl. Math.* **162** (2004), no. 1, p. 57–77.

[124] P. Langlois et F. Nativel – « Réduction et majoration de l'erreur d'arrondi en arithmétique à virgule flottante », *C. R. Acad. Sci. Paris Sér. I Math.* **327** (1998), no. 8, p. 781–786.

[125] — , « When automatic linear correction of rounding errors is exact », *C. R. Acad. Sci. Paris Sér. I Math.* **328** (1999), no. 6, p. 543–548.

[126] J. M. Lee – *Introduction to smooth manifolds*, Graduate Texts in Mathematics, vol. 218, Springer-Verlag, New York, 2003.

[127] X. S. Li, J. W. Demmel, D. H. Bailey, G. Henry, Y. Hida, J. Iskandar, W. Kahan, S. Y. Kang, A. Kapur, M. C. Martin, B. J. Thompson, T. Tung et D. J. Yoo – « Design, implementation and testing of extended and mixed precision BLAS », *ACM Trans. Math. Softw.* **28** (2002), no. 2, p. 152–205.

[128] S. Linnainmaa – « Analysis of some known methods of improving the accuracy of floating-point sums », *BIT* **14** (1974), p. 167–202.

[129] — , « Towards accurate statistical estimation of rounding errors in floating-point computations », *BIT* **15** (1975), no. 2, p. 165–173.

[130] — , « Software for doubled-precision floating-point computations », *ACM Trans. Math. Software* **7** (1981), no. 3, p. 272–283.

[131] — , « Error linearization as an effective tool for experimental analysis of the numerical stability of algorithms », *BIT* **23** (1983), no. 3, p. 346–359.

[132] D. G. Luenberger – *Optimization by vector space methods*, John Wiley & Sons Inc., New York, 1969.

[133] D. S. Mackey, N. Mackey et F. Tisseur – « Structured tools for structured matrices », *Electron. J. Linear Algebra* **10** (2003), p. 106–145 (electronic).

[134] — , « $\mathbb{G}$-reflectors : analogues of Householder transformations in scalar product spaces », *Linear Algebra Appl.* **385** (2004), p. 187–213.

[135] — , « Structured factorizations in scalar product spaces », Numerical Analysis Report No. 421, Manchester Centre for Computational Mathematics, Manchester, England, November 2004.

[136] — , « Structured mapping problems for automorphism groups, lie algebras and jordan algebras associated with scalar products », Numerical analysis report, Manchester Centre for Computational Mathematics, Manchester, England, 2004, In preparation.

[137] M. A. Malcolm – « On accurate floating-point summation », *Comm. ACM* **14** (1971), no. 11, p. 731–736.

[138] M. Mignotte – *Mathematiques pour le calcul formel*, Presses Universitaires de France, Paris, 1989.

[139] O. Møller – « Note on quasi double-precision », *BIT* **5** (1965), p. 251–255.

[140] — , « Quasi double-precision in floating point addition », *BIT* **5** (1965), p. 37–50.

[141] R. E. Moore – *Interval analysis*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1966.

[142] — , *Methods and applications of interval analysis*, SIAM Studies in Applied Mathematics, vol. 2, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, Pa., 1979.

[143] R. G. Mosier – « Root neighborhoods of a polynomial », *Math. Comp.* **47** (1986), no. 175, p. 265–273.

[144] A. Neumaier – *Interval methods for systems of equations*, Encyclopedia of Mathematics and its Applications, vol. 37, Cambridge University Press, Cambridge, 1990.

[145] — , *Introduction to numerical analysis*, Cambridge University Press, Cambridge, 2001.

[146] Y. Nievergelt – « Scalar fused multiply-add instructions produce floating-point matrix arithmetic provably accurate to the penultimate digit », *ACM Trans. Math. Software* **29** (2003), no. 1, p. 27–48.

[147] — , « Analysis and applications of priest's distillation », *ACM Trans. Math. Softw.* **30** (2004), no. 4, p. 402–433.

[148] M.-T. Noda et T. Sasaki – « Approximate GCD and its application to ill-conditioned algebraic equations », *Proceedings of the International Symposium on Computational Mathematics (Matsuyama, 1990)*, vol. 38, 1991, p. 335–351.

[149] M.-a. Ochi, M.-T. Noda et T. Sasaki – « Approximate greatest common divisor of multivariate polynomials and its application to ill-conditioned systems of algebraic equations », *J. Inform. Process.* **14** (1991), no. 3, p. 292–300.

[150] T. Ogita, S. Oishi et Y. Ushiro – « Fast verification of solutions for sparse monotone matrix equations », Topics in numerical analysis, Comput. Suppl., vol. 15, Springer, Vienna, 2001, p. 175–187.

[151] — , « Computation of sharp rigorous componentwise error bounds for the approximate solutions of systems of linear equations », *Reliab. Comput.* **9** (2003), no. 3, p. 229–239.

[152] — , « Fast inclusion and residual iteration for solutions of matrix equations », Inclusion methods for nonlinear problems with applications in engineering, economics and physics (Munich, 2000), Comput. Suppl., vol. 16, Springer, Vienna, 2003, p. 171–184.

[153] T. Ogita, S. M. Rump et S. Oishi – « Accurate sum and dot product », *SIAM J. Sci. Comput.* **26** (2005), no. 6, p. 1955–1988.

[154] — , « Verified solution of linear systems without directed rounding », Technical Report No. 2005-04, Advanced Research Institute for Science and Engineering, Waseda University, 2005.

[155] S. Oishi – « Fast enclosure of matrix eigenvalues and singular values via rounding mode controlled computation », *Linear Algebra Appl.* **324** (2001), no. 1-3, p. 133–146, Special issue on linear algebra in self-validating methods.

[156] S. Oishi et S. M. Rump – « Fast verification of solutions of matrix equations », *Numer. Math.* **90** (2002), no. 4, p. 755–773.

[157] A. M. Ostrowski – *Solution of equations and systems of equations*, Academic Press, New York, 1966.

[158] M. L. Overton – *Numerical computing with IEEE floating point arithmetic*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001, Including one theorem, one rule of thumb, and one hundred and one exercises.

[159] V. Y. Pan – « Numerical computation of a polynomial gcd and extensions », Rapport de Recherche 2969, Institut National de Recherche en Informatique et en Automatique (INRIA), Août 1996.

[160] — , « Solving a polynomial equation : some history and recent progress », *SIAM Rev.* **39** (1997), no. 2, p. 187–220.

[161] — , « Computation of approximate polynomial GCDs and an extension », *Inform. and Comput.* **167** (2001), no. 2, p. 71–85.

[162] M. Pichat – « Correction d'une somme en arithmétique à virgule flottante », *Numer. Math.* **19** (1972), p. 400–406.

[163] — , « Contributions à l'etude des erreurs d'arrondi en arithmétique à virgule flottante », Thèse, Université Scientifique et Médicale de Grenoble, Grenoble, France, 1976.

[164] D. M. Priest – « Algorithms for arbitrary precision floating point arithmetic », *Proceedings of the 10th IEEE Symposium on Computer Arithmetic (Arith-10)* (Grenoble, France) (P. Kornerup et D. W. Matula, éds.), IEEE Computer Society Press, Los Alamitos, CA, 1991, p. 132–144.

[165] — , « On properties of floating point arithmetics : Numerical stability and the cost of accurate computations », Thèse, Mathematics Department, University of California, Berkeley, CA, USA, November 1992, `ftp://ftp.icsi.berkeley.edu/pub/theory/priest-thesis.ps.Z`, p. 126.

[166] L. Qiu, A. L. Tits et Y. G. Yang – « On the computation of the real Hurwitz-stability radius », *IEEE Trans. Automat. Control* **40** (1995), no. 8, p. 1475–1476.

[167] N. Revol – « Arithmétique par intervalles », *Réseaux et Systèmes Répartis, Calculateurs Parallèles* **13** (2001), no. 4-5, p. 387–426.

[168] J. R. Rice – « A theory of condition », *SIAM J. Numer. Anal.* **3** (1966), p. 287–310.

[169] D. R. Ross – « Reducing truncation errors using cascading accumulators », *J. Assoc. Comput. Mach.* **8** (1965), no. 1, p. 32–33.

[170] S. M. Rump – « Fast and parallel interval arithmetic », *BIT* **39** (1999), no. 3, p. 534–554.

[171] — , « INTLAB — INTerval LABoratory », Developments in Reliable Computing (T. Csendes, éd.), Kluwer, Dordrecht, Netherlands, 1999, p. 77–105.

[172] — , « Structured perturbations. I. Normwise distances », *SIAM J. Matrix Anal. Appl.* **25** (2003), no. 1, p. 1–30 (electronic).

[173] — , « Structured perturbations. II. Componentwise distances », *SIAM J. Matrix Anal. Appl.* **25** (2003), no. 1, p. 31–56 (electronic).

[174] — , « Eigenvalues, pseudospectrum and structured perturbations », *Linear Algebra Appl.* (2005), To appear.

[175] D. Rupprecht – « Élément de géométrie approchée : étude du pgcd et de la factorisation », Thèse, Université de Nice-Sophia Antipolis, Janvier 2000.

[176] T. Sasaki et M.-T. Noda – « Approximate square-free decomposition and root-finding of ill-conditioned algebraic equations », *J. Inform. Process.* **12** (1989), no. 2, p. 159–168.

[177] R. Schätzle – « On the perturbation of the zeros of complex polynomials », *IMA J. Numer. Anal.* **20** (2000), no. 2, p. 185–202.

[178] A. Schönhage – « Quasi-gcd computations », *J. Complexity* **1** (1985), no. 1, p. 118–137.

[179] J. R. Shewchuk – « Robust adaptive floating-point geometric predicates », *SCG '96 : Proceedings of the twelfth annual symposium on Computational geometry* (New York, NY, USA), ACM Press, 1996, p. 141–150.

[180] — , « Adaptive precision floating-point arithmetic and fast robust geometric predicates », *Discrete Comput. Geom.* **18** (1997), no. 3, p. 305–363, ACM Symposium on Computational Geometry (Philadelphia, PA, 1996).

[181] J. Sreedhar, P. Van Dooren et A. L. Tits – « A fast algorithm to compute the real structured stability radius », Stability theory (Ascona, 1995), Internat. Ser. Numer. Math., vol. 121, Birkhäuser, Basel, 1996, p. 219–230.

[182] P. H. Sterbenz – *Floating-point computation*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1974, Prentice-Hall Series in Automatic Computation.

[183] H. J. Stetter – « Analysis of zero clusters in multivariate polynomial systems », *ISSAC '96 : Proceedings of the 1996 international symposium on Symbolic and algebraic computation*, ACM Press, 1996, p. 127–136.

[184] — , « The nearest polynomial with a given zero, and similar problems », *SIGSAM Bulletin (ACM Special Interest Group on Symbolic and Algebraic Manipulation)* **33** (1999), no. 4, p. 2–4.

[185] — , « Polynomials with coefficients of limited accuracy », Computer algebra in scientific computing—CASC'99 (Munich), Springer, Berlin, 1999, p. 409–430.

[186] — , *Numerical polynomial algebra*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2004.

[187] J. guang Sun – « Optimal backward perturbation bounds for linear systems and linear least squares problems », Report UMINF 96-15, Department of Computing Science, University of Umeå, Sweden, December 1996.

[188] M. Tienari – « A statistical model of roundoff error for varying length floating-point arithmetic », *BIT* **10** (1970), p. 355–365.

[189] F. Tisseur et S. Graillat – « Structured condition numbers and backward errors in scalar product spaces », Numerical Analysis Report No. 473, Manchester Centre for Computational Mathematics, Manchester, England, 2005.

[190] F. Tisseur – « Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems », *SIAM J. Matrix Anal. Appl.* **22** (2001), no. 4, p. 1038–1057 (electronic).

[191] F. Tisseur et N. J. Higham – « Structured pseudospectra for polynomial eigenvalue problems, with applications », *SIAM J. Matrix Anal. Appl.* **23** (2001), no. 1, p. 187–208 (electronic).

[192] K.-C. Toh et L. N. Trefethen – « Pseudozeros of polynomials and pseudospectra of companion matrices », *Numer. Math.* **68** (1994), no. 3, p. 403–425.

[193] L. N. Trefethen – « Pseudospectra of matrices », Numerical analysis 1991 (Dundee, 1991), Pitman Res. Notes Math. Ser., vol. 260, Longman Sci. Tech., Harlow, 1992, p. 234–266.

[194] — , « Computation of pseudospectra », Acta numerica, 1999, Cambridge Univ. Press, Cambridge, 1999, p. 247–295.

[195] — , « Computation of pseudospectra », Acta numerica, 1999, Acta Numer., vol. 8, Cambridge Univ. Press, Cambridge, 1999, p. 247–295.

[196] L. N. Trefethen et M. Embree – *Spectra and pseudospectra*, Princeton University Press, Princeton, NJ, 2005, The behavior of nonnormal matrices and operators.

[197] C. W. Ueberhuber – *Numerical computation. 1*, Springer-Verlag, Berlin, 1997, Methods, software, and analysis, Translated and revised from the 1995 German original.

[198] J. H. Wilkinson – *Rounding errors in algebraic processes*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1963.

[199] — , « The perfidious polynomial », Studies in numerical analysis, MAA Stud. Math., vol. 24, Math. Assoc. America, Washington, DC, 1984, p. 1–28.

[200] J. M. Wolfe – « Reducing truncation errors by programming », *Comm. ACM* **7** (1964), no. 6, p. 355–356.

[201] Z. Zeng – « A method computing multiple roots of inexact polynomials », *Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation* (New York), ACM, 2003, p. 266–272 (electronic).

[202] — , « Algorithm 835 : MultRoot—a Matlab package for computing polynomial roots and multiplicities », *ACM Trans. Math. Softw.* **30** (2004), no. 2, p. 218–236.

[203] — , « The approximate GCD of inexact polynomials. Part I : a univariate algorithm », Preprint, Northeastern Illinois University, 2004, http://www.neiu.edu/~zzeng/.

[204] — , « Computing multiple roots of inexact polynomials », *Math. Comp.* **74** (2005), no. 250, p. 869–903 (electronic).

[205] Z. Zeng et B. H. Dayton – « The approximate GCD of inexact polynomials. Part II : a multivariate algorithm », *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation* (New York), ACM, 2004, p. 320–327 (electronic).

[206] H. Zhang – « Numerical condition of polynomials in different forms », *ETNA* **12** (2001), p. 66–87.

[207] R. Zippel – *Effective polynomial computation.*, The Kluwer International Series in Engineering and Computer Science. 241. Dordrecht : Kluwer Academic Publ., 1993 (English).

Résumé :

Les travaux présentés dans cette thèse portent sur la stabilité et la précision de certains algorithmes numériques. Les contributions de cette thèse se situent à quatre niveaux : 1) Amélioration de la précision : on propose un algorithme de Horner compensé qui calcule un résultat avec la même précision que s'il avait été calculé par le schéma de Horner classique mais avec une précision interne doublée. 2) Applications des pseudozéros : on propose des applications des pseudozéros en calcul formel (primalité approchée) et en théorie du contrôle (rayon de stabilité et pseudoabscisse). 3) Prise en compte des perturbations réelles : on donne des formules calculables pour le conditionnement réel et l'erreur inverse réelle pour les problèmes de l'évaluation polynomiale et le calcul de racines. Nous montrons qu'il y a peu de différences entre le conditionnement réel et le conditionnement classique. Néanmoins, nous montrons que l'erreur inverse réelle peut être significativement plus grande que l'erreur inverse classique. 4) Perturbations matricielles structurées : Nous étudions la notion de pseudospectre structuré pour les matrices Toeplitz, circulantes et Hankel. Nous montrons qu'il n'y a pas de différence entre le pseudospectre structuré et le pseudospectre classique. Nous étudions aussi des conditionnements pour les systèmes linéaires et l'inversion matricielle pour des structures dérivant d'algèbres de Lie et de Jordan. Nous étudions pour quelles sous-classes de ces structures il n'y a pas ou peu de différences entre les conditionnements structurés et non structurés.

Mots-clés :

Arithmétique des ordinateurs, analyse d'erreur, arithmétique d'intervalle, algèbre linéaire numérique, analyse numérique, calcul formel, algèbre polynomiale numérique.

Abstract:

The results summarized in the document deal with the stabiliy and accuracy of some numerical algorithms. The contributions of this work are divided into four levels : 1) Improvement of the accuracy: we present a compensated Horner scheme that computes a result as if computed in twice the working precision. 2) Applications of pseudozero set: we propose some applications of pseudozeros in computer algebra (approximate coprimeness) and in control theory (stability radius and pseudoabscissa). 3) Real perturbations: we give computable formulas for the real condition number and real backward error for the problem of polynomial evaluation and the computation of zeros. We show that there is little difference between the real and complex condition numbers. On the contrary, we show that the real backward error can be significantly larger than the complex one. 4) Structured matrix perturbations: We study the notion of structured pseudospectra for Toeplitz, Hankel and circulant matrices. We show for this structures there is no difference between the structured and the unstructured pseudospectra. We also study structured condition number for linear systems, inversions and distance to singularity for structures deriving from Lie and Jordan algebras. We show that under mild assumptions there is little or no difference between the structured and the unstructured condition numbers.

Keywords:

Computer arithmetic, rounding error analysis, interval arithmetic, numerical linear algebra, numerical analysis, computer algebra, numerical polynomial algebra.