HAL open science

Un peu de théorie des nombres et de calcul formel au service de l'arithmétique des ordinateurs

Nicolas Brisebarre

To cite this version:

Nicolas Brisebarre. Un peu de théorie des nombres et de calcul formel au service de l'arithmétique des ordinateurs. Arithmétique des ordinateurs. École Normale Supérieure de Lyon, 2017. tel-01658342v1

HAL Id: tel-01658342

https://hal.science/tel-01658342v1

Submitted on 7 Dec 2017 (v1), last revised 3 Jan 2019 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Un peu de théorie des nombres et de calcul formel au service de l'arithmétique des ordinateurs

Mémoire d'habilitation à diriger les recherches

présenté le 5 septembre 2017 à

l'École Normale Supérieure de Lyon

par

Nicolas Brisebarre

Chargé de recherche au CNRS

SPÉCIALITÉ : INFORMATIQUE

après avis des rapporteurs :

LAURENT BARATCHART, Directeur de recherche INRIA, INRIA Sophia Antipolis
VALÉRIE BERTHÉ, Directrice de recherche CNRS à l'IRIF, Université Paris Diderot
BRIGITTE VALLÉE, Directrice de recherche CNRS au GREYC, Université de Caen

devant le jury formé de :

LAURENT BARATCHART, Directeur de recherche INRIA, INRIA Sophia Antipolis
VALÉRIE BERTHÉ, Directrice de recherche CNRS à l'IRIF, Université Paris Diderot
JEAN-MICHEL MULLER, Directeur de recherche CNRS au LIP, ÉNS Lyon
BRIGITTE VALLÉE, Directrice de recherche CNRS au GREYC, Université de Caen
ALAIN YGER, Professeur à l'Université de Bordeaux

Table des matières

Introduction **7**

1 Un petit peu d'arithmétique virgule flottante **9**

2 Quelques rappels d'algorithmique des réseaux euclidiens **11**
2.1 Notion de réseau . . . 11
2.2 Invariants d'un réseau . . . 12
2.2.1 Déterminant d'un réseau . . . 13
2.2.2 Minima successifs . . . 13
2.3 Notion de base réduite . . . 14
2.3.1 Orthogonalisation de Gram-Schmidt . . . 14
2.3.2 Défaut de longueur, défaut d'orthogonalité . . . 15
2.4 Diverses notions de réduction . . . 15
2.5 Problèmes algorithmiques . . . 17
2.5.1 Problème du plus court vecteur non nul . . . 17
2.5.2 Problème du vecteur le plus proche . . . 17
2.6 Algorithmes de réduction de réseaux . . . 17
2.6.1 L'algorithme de Lenstra, Lenstra et Lovász . . . 17
2.6.2 Les algorithmes BKZ et HKZ . . . 18
2.6.3 Implémentations . . . 18
2.7 Algorithmes de résolution de CVP . . . 19
2.7.1 L'algorithme du plan le plus proche de Babai . . . 19
2.7.2 L'algorithme du plongement de Kannan . . . 19

3 Quelques rappels de théorie de l'approximation **21**
3.1 Approximations polynomiales . . . 21
3.1.1 Densité des polynômes dans $(\mathcal{C}([a,b]), \|.\|_\infty)$ . . . 22
3.1.2 Approximation minimax . . . 22
3.2 Interpolation et approximation . . . 25
3.2.1 Polynômes de Tchebychev . . . 26
3.3 Polynômes orthogonaux - Séries de Tchebychev . . . 28
3.3.1 Polynômes orthogonaux . . . 28
3.3.2 Constante de Lebesgue . . . 30

I Outils pour l'évaluation efficace et certifiée de fonctions **33**

Préambule **36**

4 Approximation efficace en machine **37**
4.1 Formulation générale du problème . . . 38
4.2 Approche via la réduction de réseaux euclidiens . . . 40
4.3 Résolution (approchée) du CVP_∞ . . . 41
4.4 Une synthèse de notre approche sous forme de pseudo-code . . . 41
4.5 Approximation polynomiale sous contraintes sur les coefficients . . . 41
4.5.1 Introduction . . . 43
4.5.2 Déclinaison de notre approche via les réseaux euclidiens . . . 45
4.5.3 Mise en œuvre pratique . . . 47
4.5.4 Approximation polynomiale avec coefficients contraints : le cas de la norme L^2 . . . 48
4.6 Synthèse de filtres RIF . . . 49
4.6.1 Introduction . . . 49
4.6.2 Quantification des filtres FIR sous forme directe . . . 50
4.6.3 Discrétisation du problème . . . 52
4.6.4 Résultats expérimentaux . . . 53
4.6.5 Pourquoi cela fonctionne t-il si bien ? . . . 56
4.7 Approximation rationnelle efficace en virgule fixe : la E-méthode . . . 57
4.8 Perspectives . . . 57

5 Fonctions élémentaires et arrondi correct - Dilemme du fabricant de tables **59**
5.1 Introduction . . . 60
5.1.1 Cadre arithmétique - arrondi correct . . . 60
5.1.2 Implémentation de fonctions correctement arrondies : la stratégie de Ziv . . . 61
5.1.3 Évaluation correctement arrondie rapide et peu coûteuse dans le format binary64 62
5.1.4 Une approche probabiliste heuristique . . . 62
5.1.5 Interprétation (encore plus heuristique) via l'approximation diophantienne . . . 63
5.2 Formalisation des problèmes . . . 65
5.2.1 Décompte des mauvais cas . . . 65
5.2.2 Dilemme du fabricant de tables . . . 65
5.3 Sommes d'exponentielles et proportion de mauvais cas pour l'arrondi correct . . . 66
5.3.1 Estimations à l'aide de techniques de sommes d'exponentielles . . . 67
5.3.2 Une estimation du nombre de « mauvais cas » pour la première phase rapide de la stratégie de Ziv en deux étapes . . . 69
5.3.3 La fonction $\sqrt[3]{\cdot}$ . . . 70
5.3.4 La fonction $\exp$ . . . 70
5.4 Bornes pour certaines fonctions algébriques . . . 71
5.4.1 Quelques heuristiques et un résultat sur $\mu(p)$ . . . 72
5.4.2 Fonctions puissance . . . 74
5.5 Perspectives . . . 76

6 Approximations polynomiales rigoureuses **79**
6.1 Introduction . . . 80
6.2 Arithmétique d'intervalles, analyse d'intervalles . . . 81
6.2.1 Arithmétique d'intervalles . . . 81
6.2.2 Fonctions d'intervalles . . . 83
6.2.3 Le phénomène de dépendance . . . 83
6.3 Modèles de Taylor . . . 83
6.3.1 Remplacer les polynômes de Taylor par des approximations plus fines . . . 84
6.4 Modèles de Tchebychev . . . 85
6.4.1 Opérations sur les polynômes dans la base de Tchebychev . . . 85
6.4.2 Fonctions de base . . . 86
6.4.3 Addition et multiplication . . . 87
6.4.4 Composition . . . 88
6.4.5 Croissance des coefficients et surestimation . . . 89

6.5 Résultats expérimentaux . . . 89
6.6 Approximations polynomiales rigoureuses : preuve formelle . . . 90
6.7 Perspectives . . . 90
6.7.1 Méthodes spectrales certifiées . . . 90
6.7.2 Arithmétique rapide pour les approximations polynomiales rigoureuses . . . 91
6.7.3 Validation formelle des approximations . . . 91
6.7.4 Cas des équations différentielles non-linéaires . . . 91

II En amont : algorithmes bas niveau en arithmétique des ordinateurs, approximation diophantienne 93

7 Algorithmes bas niveau en arithmétique des ordinateurs 95
7.1 Fonctions calculables à l'aide d'une instruction fused multiply-and-add . . . 95
7.2 Conversions binaire/décimal rapides et avec arrondi correct, comparaisons binaire/décimal 96
7.3 Algorithmes et implantations matérielles efficaces pour la cryptographie : cryptographie fondée sur les couplages . . . 96

8 Approximation diophantienne 99
8.1 Résolution effective de systèmes d'équations aux différences . . . 99
8.2 Détermination explicites de mesures d'irrationalité de certaines constantes . . . 100
8.3 Fonctions modulaires . . . 100
8.4 Corps de séries formelles . . . 100

9 Une synthèse de mes perspectives de recherche 103
9.1 Approximation efficace en machine . . . 104
9.1.1 Synthèse de filtres RIF et RII . . . 104
9.2 Fonctions élémentaires et arrondi correct - Dilemme du fabricant de tables . . . 104
9.3 Approximations polynomiales rigoureuses . . . 104
9.3.1 Méthodes spectrales certifiées . . . 104
9.3.2 Arithmétique rapide pour les approximations polynomiales rigoureuses . . . 105
9.3.3 Validation formelle des approximations . . . 105
9.3.4 Cas des équations différentielles non-linéaires . . . 105
9.3.5 Applications . . . 105
9.4 Vers une bibliothèque de calcul à trois niveaux de sécurité . . . 106

A Carrière et formation 107

B Liste de publications 109

C Encadrement d'étudiants 113

Introduction

Mon travail porte sur certains aspects de l'arithmétique des ordinateurs qui est un des thèmes principaux de l'équipe AriC (anciennement Arénaire) du LIP, le laboratoire d'informatique de l'ÉNS Lyon. J'évolue au sein de cette équipe depuis quinze ans maintenant. Un des objectifs généraux que l'on y poursuit est l'amélioration de la qualité (précision, fiabilité, garantie), de la vitesse et du coût (en mémoire, en énergie, en silicium) des calculs en machine. On souhaite pouvoir ainsi produire des logiciels fiables, rapides et portables et du matériel (processeurs, systèmes embarqués par exemple) rapide, fiable et moins coûteux.

Une caractéristique de mon approche de ces questions d'arithmétique des ordinateurs est la mobilisation d'outils de théorie des nombres (plus précisément d'approximation diophantienne) et de calcul formel (plus précisément de calcul symbolique-numérique) pour modéliser et attaquer ces problèmes.

Après des travaux de mathématiques, avec une coloration algorithmique et effective, sur des problèmes de détermination de solutions de certains systèmes d'équations aux différences, de calculs de mesures d'irrationalité de certaines constantes et d'estimation de la taille des coefficients de Fourier de fonctions modulaires, j'ai entamé en 2002 des recherches en arithmétique des ordinateurs en réfléchissant, pour commencer, à des problèmes de théorie des nombres qui y apparaissent.

Je me suis rapidement intéressé à ce qui concerne l'évaluation des fonctions élémentaires, qui est devenu mon sujet d'intérêt principal. En collaboration avec plusieurs collègues, j'ai procédé à une étude de fond de la plupart des étapes intervenant dans ce processus de synthèse logicielle ou matérielle de bibliothèques d'évaluation de fonctions en précision finie. À ma connaissance, elles constituent l'état de l'art. Je me passionne en particulier pour des questions d'approximation de fonctions, que ce soit en privilégiant l'efficacité du calcul de l'approximation ou celle de la qualité de l'évaluation de l'approximation ou encore la certification de l'erreur d'approximation. Je présente mes résultats dans la première (et principale) partie de ce mémoire. Les sujets des thèses que j'ai co-encadrées relèvent tous de cette première partie.

La seconde partie sera consacrée à l'évocation rapide de

— la mise au point d'algorithmes de bas niveau pour accélérer certaines primitives pour le calcul flottant ou la cryptologie à clé publique (couplages sur des courbes elliptiques).
— travaux, plus ou moins anciens, en théorie des nombres et plus précisément en approximation diophantienne.

Enfin, je donne en annexe un bref rappel de mon parcours professionnel et de ma formation, une liste de publications à jour et je présente un peu les thèses et le post-doctorat que j'ai encadrés ou co-encadrés. L'encadrement d'étudiants est pour moi une partie cardinale de mon activité et mon travail est influencé en profondeur par l'interaction avec mes étudiants.

Les citations bibliographiques du type [n] renvoient à la liste de références en fin de texte et celles du type **J-n, C-n, ...** renvoient à la liste de publications en Annexe B.

Chapitre 1

Un petit peu d'arithmétique virgule flottante

Avant de démarrer notre promenade, il nous faut rappeler quels sont les nombres avec lesquels nous allons travailler. Il y a diverses façons de représenter les nombres réels en machine mais le meilleur compromis à ce jour est celui offert par l'arithmétique virgule flottante. Le livre [156] lui est dédié et il constituera notre référence principale pour toute question relative aux nombres flottants.

Définition 1.1. *Un nombre flottant en base β ($\beta \in \mathbb{N}$, $\beta \geqslant 2$) et précision p ($p \in \mathbb{N}$, $p \geqslant 2$), est un nombre de la forme*

$$x = (-1)^s \frac{M}{\beta^{p-1}} \cdot \beta^E,$$

où :

- *E, appelé l'*exposant*, est un entier relatif tel que $E_{min} \leqslant E \leqslant E_{max}$, les exposants $E_{min} < 0 < E_{max}$ étant fixés ;*
- *M est un entier naturel, appelé mantisse entière, représenté en base β, $\beta^{p-1} \leqslant |M| \leqslant \beta^p - 1$.*
- *$s \in \{0, 1\}$ est le bit de signe de x.*

Le nombre zéro est un cas spécial, cf. [156, Chap. 3], que l'on adjoint à l'ensemble des nombres flottants en précision p. Cela donne un ensemble que l'on note $\mathcal{F}_p$.

Par exemple, en base 2, avec les bornes requises sur l'exposant E :

- $\mathcal{F}_{24}$ est l'ensemble des flottants au format binary32 (anciennement single precision)
- $\mathcal{F}_{53}$ est l'ensemble des flottants au format binary64 (anciennement double precision)
- $\mathcal{F}_{113}$ est l'ensemble des flottants au format binary128 (anciennement quadruple precision)

Remarque 1.2. Cette définition est en fait celle d'un nombre flottant normal. Pour simplifier l'exposition, j'ai choisi de ne pas parler des flottants dénormalisés puisqu'ils n'apparaîtront pas (ou marginalement) dans le mémoire. On peut consulter [156, Chap. 2.1] pour la définition usuelle.

La base 2 est la plus commune et c'est avec elle que nous travaillerons dans tout ce texte. La base 10 est aussi utilisée : c'est la base dans laquelle nous comptons et elle est utilisée pour les calculs financiers et dans les calculatrices de poche. Le système de calcul formel Maple utilise lui aussi la base 10 pour la représentation interne des nombres.

En 1985, le standard IEEE 754 [37, 3] pour l'arithmétique virgule flottante en base 2 a été publié, suivi un peu plus tard par sa généralisation [36, 4] à l'arithmétique décimale, le standard IEEE 854. Cela a contribué de façon clé à mettre fin à un grand désordre : les progrès en terme de précision, fiabilité et portabilité des calculs en machine ont été très substantiels. Ce standard spécifie notamment divers formats, le comportement des opérations arithmétiques de base ($+, -, \times, \div$ et $\sqrt{}$) ou les conversions. Actuellement, la plupart des systèmes du commerce suivent les règles du standard. Une révision de ce standard, appelée IEEE 754-2008 (qui inclut aussi l'arithmétique virgule flottante en base 10), a été adoptée en juin 2008 [95]. La table 1.1 donne les principaux paramètres des formats en base 2 de cette révision.

	précision p	exposant minimal E_{min}	exposant maximal E_{max}
binary32 (simple)	24	-126	127
binary64 (double)	53	-1022	1023
binary128 (quadruple)	113	-16382	16383

TABLE 1.1 – Principaux paramètres des formats binaires (jusqu'à 128 bits) spécifiés par le standard [95].

En général, le résultat d'une opération (ou d'une fonction) sur des nombres flottants n'est pas représentable exactement dans l'arithmétique virgule flottante utilisée. Il y a donc besoin de procéder à un arrondi, une opération clé en arithmétique flottante. Les quatre modes d'arrondi présents dans le standard IEEE 754-2008 standard sont :

— l'arrondi vers $-\infty$: $\mathrm{RD}(x)$ est la plus grande valeur (qui peut être soit un nombre flottant soit $-\infty$) inférieure ou égale à x ;
— l'arrondi vers $+\infty$: $\mathrm{RU}(x)$ est la plus petite valeur (qui peut être soit un nombre flottant soit $+\infty$) supérieure ou égale à x ;
— arrondi vers zéro : $\mathrm{RZ}(x)$ est le nombre flottant le plus proche de x dont la valeur absolue est inférieure ou égale à la valeur absolue de x (il vaut $\mathrm{RD}(x)$ si $x \geqslant 0$ et $\mathrm{RU}(x)$ si $x \leqslant 0$) ;
— arrondi au plus près : $\mathrm{RN}(x)$ est le nombre le plus proche de x. Il faut choisir une règle pour gérer les cas où x est exactement le milieu de deux nombres flottants consécutifs et on suit alors, en général, la règle dite de « l'arrondi pair » : x est arrondi vers le flottant dont la mantisse entière est paire. C'est le mode par défaut du standard IEEE 754-2008.

Les trois premiers modes sont appelés arrondis dirigés.

Quand on établit des propriétés sur les nombres flottants, il est assez naturel et commode de traiter ensemble les flottants ayant même exposant.

Définition 1.3. *Une binade est un intervalle de la forme* $[2^k, 2^{k+1}[$ *ou* $]-2^{k+1}, -2^k]$ *pour* $k \in \mathbb{Z}$.

Si l'arithmétique virgule flottante semble aujourd'hui le choix de représentation le plus populaire, l'arithmétique virgule fixe est encore extrêmement courante dans le monde du matériel, des systèmes embarqués du fait de sa simplicité, sa vitesse et sa facilité d'usage.

Contrairement à l'arithmétique virgule flottante, il n'y a pas de formalisation standard de l'arithmétique virgule fixe. Il y a donc de nombreuses définitions et notations pour les nombres en virgule fixe. Nous avons choisi la définition suivante, habituellement utilisée :

Définition 1.4. *Soit un facteur d'échelle réel* $s > 0$ *et des bornes entières* $m_- < m_+$, *un nombre virgule fixe est un nombre de la forme*

$$x = M \cdot s,$$

où $M \in [m_-, m_+] \cap \mathbb{Z}$ *est appelée la* mantisse entière *de* x.

Dans de nombreux cas, $s = 2^{-\ell}, \ell \in \mathbb{Z}$.

Chapitre 2

Quelques rappels d'algorithmique des réseaux euclidiens

La structure de réseau euclidien intervient dans de nombreuses questions de mathématiques, d'informatique ou de cristallographie [28, 80, 140, 39, 164, 150, 166] et je l'ai utilisée, ainsi que sa riche algorithmique, à plusieurs reprises dans mon travail. J'en donne maintenant une brève présentation, bien loin d'être exhaustive, en regroupant les énoncés, sans démonstration, qui me seront utiles dans la suite de ce mémoire. On trouvera des présentations complètes et très bien faites dans les livres [28, 80, 140, 39, 164, 213, 38] et les thèses [210, 99, 162, 198] ou encore l'article [209]. Je me suis beaucoup aidé de [209] et de [164, Chap. 2] pour rédiger cette synthèse.

2.1 Notion de réseau

Dans tout ce chapitre, $\mathbb{R}^n$ est muni de sa structure euclidienne canonique et de la mesure de Lebesgue. On notera $(\cdot|\cdot)$ le produit scalaire canonique, c.-à-d. si $x = (x_1, \ldots, x_n)$ et $y = (y_1, \ldots, y_n) \in \mathbb{R}^n$, alors

$$(x|y) = x_1 y_1 + \cdots + x_n y_n.$$

On pose $\|x\|_2 = (x|x)^{1/2} = (x_1^2 + \cdots + x_n^2)^{1/2}$ et $\|x\|_\infty = \max_{1 \leqslant i \leqslant n} |x_i|$. On notera $\mathrm{vol}(\cdot)$ la mesure de Lebesgue.

En dehors de la section 2.5 où l'on considérera une norme quelconque, la norme par défaut sera la norme euclidienne $\|\cdot\|_2$.

Définition 2.1. *Un réseau est un sous-groupe discret de $\mathbb{R}^n$ non réduit à $\{0\}$.*

Nous utiliserons plutôt comme définition celle donnée par la proposition suivante :

Proposition 2.2. *Soit L une partie non vide de $\mathbb{R}^n$, L est un réseau ssi il existe des vecteurs $b_1, \ldots, b_d$ $\mathbb{R}$-linéairement indépendants tels que*

$$L = \mathbb{Z}.b_1 \oplus \cdots \oplus \mathbb{Z}.b_d.$$

La famille $(b_1, \ldots, b_d)$ est une base du réseau.

Exemples 2.3. $\mathbb{Z}^n$, tout sous-groupe de $\mathbb{Z}^n$, l'exemple $\mathbb{Z}(2,0) \oplus \mathbb{Z}(1,2)$ de la figure 2.1.

Remarque 2.4. On dit qu'un réseau L est entier (resp. rationnel) lorsque $L \subset \mathbb{Z}^n$ (resp. $\mathbb{Q}^n$).

Un réseau euclidien n'est pas un $\mathbb{R}$-espace vectoriel mais en tant que module de type fini sur l'anneau principal $\mathbb{Z}$, il en conserve certaines bonnes propriétés, telles que la notion de dimension :

Proposition 2.5. *Si $(e_1, \ldots, e_k)$ et $(f_1, \ldots, f_j)$ sont deux familles libres sur $\mathbb{R}$ qui engendrent le même réseau, alors $k = j$ (cet entier est appelé le rang ou la dimension du réseau) et il existe une matrice U de dimension $k \times k$, à coefficients entiers, de déterminant ± 1 (on dit que U est unimodulaire) telle que $(e_i) = (f_i)U$.*

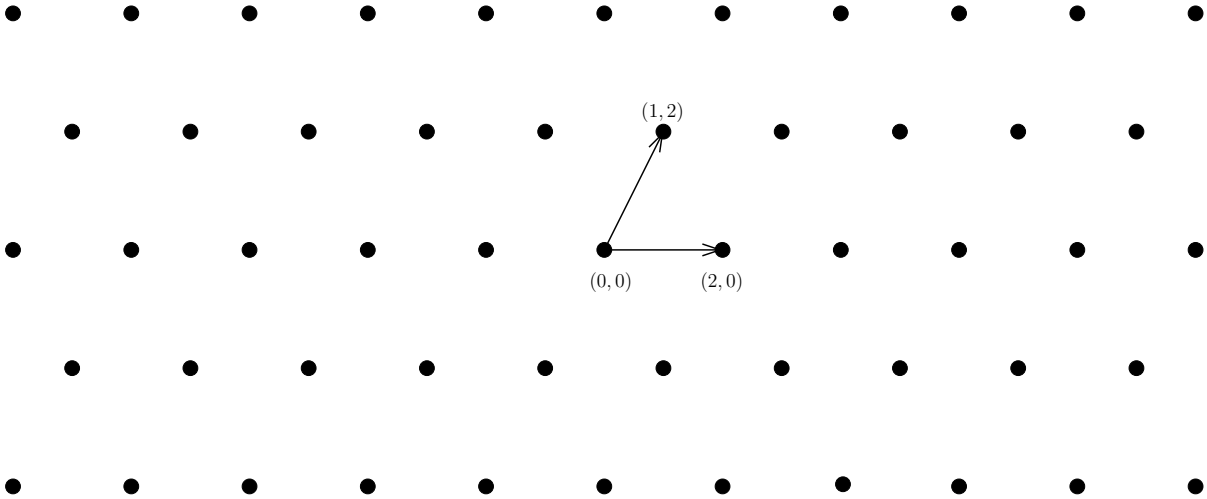


FIGURE 2.1 – Le réseau $\mathbb{Z}(2,0) \oplus \mathbb{Z}(1,2)$

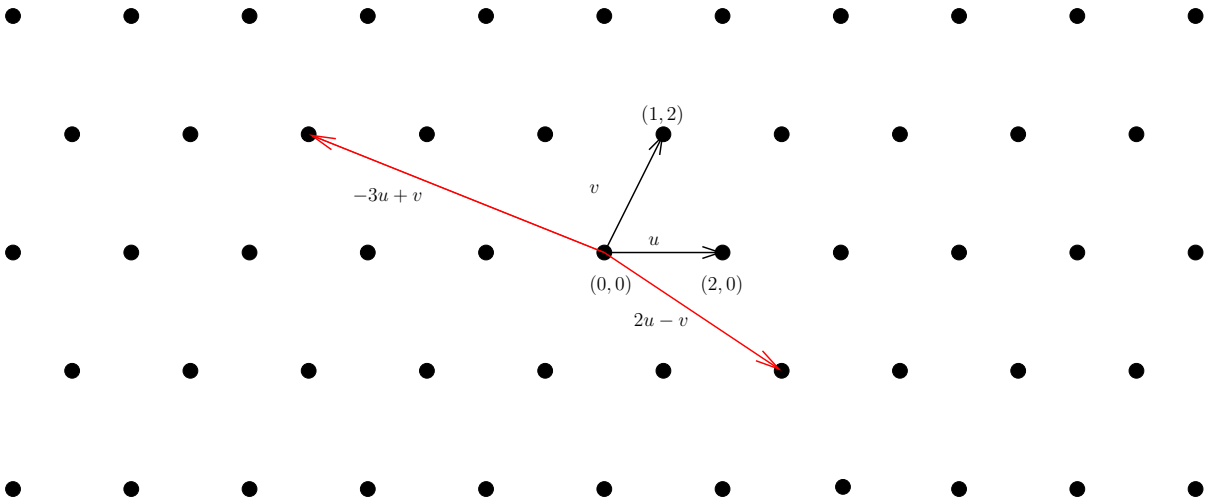


FIGURE 2.2 – Le réseau $\mathbb{Z}(2,0) \oplus \mathbb{Z}(1,2)$ et deux de ses bases (u,v) et $(2u-v, -3u+v)$

Il y a donc une infinité de bases d'un réseau donné dès que sa dimension est supérieure ou égale à 2 mais nous verrons dans la suite que certaines sont plus intéressantes que d'autres et nous allons travailler pour obtenir ces meilleures bases. Pour illustrer cette idée de qualité d'une base, examinons la figure 2.2. On dispose de la base (u,v) avec $u = (2,0)$ et $v = (1,2)$ et de la base $(2u-v, v-3u)$. Supposons un instant que l'on enlève de la figure les points noirs qui dessinent la portion du réseau visible ici. Il est alors bien plus aisé (enfin, c'est l'impression du rédacteur) d'imaginer, correctement, cette grille à partir de la base (u,v) que de la base $(2u-v, v-3u)$. Le fait que la base (u,v) soit formée de vecteurs plus orthogonaux et plus courts que $(2u-v, v-3u)$ est certainement la raison de cette observation.

2.2 Invariants d'un réseau

Outre la dimension, un réseau possède d'autres invariants, au rôle fondamental, que l'on introduit à présent.

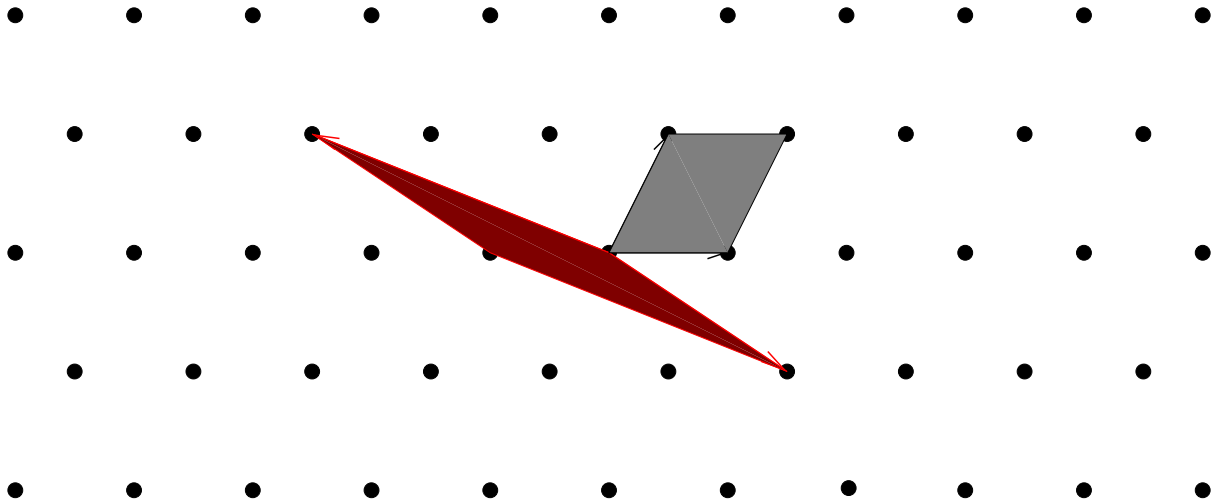


FIGURE 2.3 – Invariance du déterminant du réseau $\mathbb{Z}(2,0) \oplus \mathbb{Z}(1,2)$: les deux surfaces ont même aire

2.2.1 Déterminant d'un réseau

Définition 2.6. *Soit L un réseau de $\mathbb{R}^n$, $(b_1, \ldots, b_d)$ une base de L. On appelle discriminant de L, et on note $\Delta(L)$ ou disc(L) le déterminant de la matrice de Gram $(b_i|b_j)_{1\leqslant i,j\leqslant d}$. Le déterminant (ou volume) de L noté* dét L *est la racine carrée de $\Delta(L)$.*

Lorsque $d = n$ (on dit que L est un réseau total), on a

$$\text{dét}\, L = |\text{dét}\,(b_i)| = \text{ volume de } \left\{ \sum_{1\leqslant i\leqslant n} \alpha_i b_i, \alpha_i \in [0,1[\right\}.$$

L'ensemble $\left\{\sum_{1\leqslant i\leqslant n} \alpha_i b_i, \alpha_i \in [0,1[\right\}$ est le *parallélotope fondamental* du réseau. On illustre l'invariance de son volume dans la figure 2.3.

Remarque 2.7. Cette invariance du déterminant nous confirme que, pour une base, le fait d'avoir des vecteurs courts va de pair avec le fait d'avoir des vecteurs les plus orthogonaux possible.

Remarque 2.8. Certains auteurs, voir [39, 145] par exemple, appellent déterminant du réseau ce que nous appelons discriminant du réseau. En revanche, le terme de volume garde la même définition.

2.2.2 Minima successifs

Soit L un réseau de rang d. Comme L est discret, L admet au moins un plus court vecteur non nul. Sa longueur est le premier minimum $\lambda_1(L)$ du réseau.

Pour k entre 1 et d, le k-ème minimum de L est défini par

$$\lambda_k(L) := \inf\{\rho \in \mathbb{R}; \dim \text{Vect}_{\mathbb{R}}(L \cap B(0,\rho)) \geqslant k\},$$

où $\text{Vect}_{\mathbb{R}}(\text{A})$ désigne l'espace vectoriel sur $\mathbb{R}$ engendré par A. En d'autres termes, le k-ème minimum est le plus petit réel ρ tel qu'il existe k vecteurs $\mathbb{R}$-linéairement indépendants du réseau de norme au plus ρ.

Le premier théorème de Minkovski nous donne $\lambda_1(L) \leqslant \sqrt{d}(\text{dét}\,(L))^{1/d}$. Ce résultat se généralise dans l'énoncé suivant :

Théorème 2.9. (Second théorème de Minkovski) *Pour tout réseau L de rang d, pour tout $r \leqslant d$, on a*

$$\prod_{i=1}^{r} \lambda_i(L) \leqslant d^{r/2}(\text{dét}\, L)^{r/d}. \tag{2.1}$$

Pour $r = d$, on a

$$1 \leqslant \frac{\prod_{i=1}^{d} \lambda_i(L)}{\text{dét}\, L} \leqslant d^{d/2}. \tag{2.2}$$

Ce résultat est remarquable car il relie, de manière non grossière, des objets difficiles à calculer (les minima successifs) à une quantité facilement accessible, le volume du réseau correspondant.

Remarque 2.10. Cet énoncé est une version un peu affaiblie du résultat de Minkowski : on peut en fait remplacer le facteur $d^{r/2}$ dans (2.1) et le facteur $d^{d/2}$ dans (2.2) par des facteurs plus petits, dépendant d'une expression appelée invariant d'Hermite, cf. [209][164, Chap. 2].

2.3 Notion de base réduite

Idéalement, on a envie d'avoir une base formée des vecteurs les plus courts possibles.

Proposition 2.11. *Soit L un réseau de rang $\leqslant 4$, il existe une base formée de vecteurs réalisant les minima successifs de L.*

On sait de plus les calculer efficacement à l'aide de l'algorithme de Lagrange [119], plus connu sous le nom d'algorithme de Gauss, et de son extension à la dimension 3 [210] et à la dimension 4 [189].

Malheureusement, en dimension $\geqslant 5$, il n'est pas toujours possible de trouver une base formée de vecteurs réalisant les minima successifs, comme l'illustre cet exemple, mis en évidence par Korkine et Zolotarev.

Soit L le réseau donné (les vecteurs de la base sont les colonnes) par

$$\begin{pmatrix} 2 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Il n'est pas difficile de montrer que les minima successifs, tous égaux à 2, sont réalisés par

$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

qui ne forment pas une base du réseau L.

Cela nous amène à essayer de définir des notions et des niveaux de qualité d'une base, ce que l'on fait via le concept de réduction. Bien sûr, la difficulté de calcul de la base est aussi intégrée à cette notion de réduction.

Comme nous l'avons déjà vu, il est agréable de disposer d'une base formée de vecteurs aussi orthogonaux que possible et l'on va mesurer l'orthogonalité d'une base d'un réseau en la comparant à la base orthogonalisée correspondante.

2.3.1 Orthogonalisation de Gram-Schmidt

On rappelle dans l'algorithme 1 le procédé, bien classique, d'orthogonalisation de Gram-Schmidt dans $\mathbb{R}^n$.

Remarques 2.12. Si $(b_1, \ldots, b_d)$ est une base d'un réseau L, alors
— on a dét $L = \prod_{1 \leqslant i \leqslant n} \|b_i^*\|$.
— pour tout $i = 1, \ldots, d$, $\|b_i^*\| \leqslant \|b_i\|$.

Algorithme 1 Orthogonalisation de Gram-Schmidt

Entrée : Une famille libre (b_i) de $\mathbb{R}^n$
Sortie : Une famille libre (b_i^*) vérifiant
— (b_i^*) est orthogonale ;
— $\mathrm{Vect}(b_1^*, \ldots, b_k^*) = \mathrm{Vect}(b_1, \ldots, b_k)$ pour tout k.

1: $b_1^* \leftarrow b_1$
2: $k \leftarrow 2$
// Pour tout k, b_k^* est le projeté de b_k orthogonalement
// au sous-espace $\mathrm{Vect}_{\mathbb{R}}(b_1, \ldots, b_{k-1})$.
// On cherche b_k^* sous la forme $b_k - \sum_{i=1}^{k-1} m_{ki} b_i^*$.
3: **tant que** $k \leqslant d$ **faire**
4: **pour** $i = 1$ à $k-1$ **faire**
5: $m_{ki} \leftarrow (b_k | b_i^*) / \|b_i^*\|^2$.
6: **fin pour**
7: $b_k^* \leftarrow b_k - \sum_{i=1}^{k-1} m_{ki} b_i^*$.
8: $k \leftarrow k+1$
9: **fin du tant que**

Lemme 2.13. *Si $(b_1, \ldots, b_d)$ est une base d'un réseau L, alors, pour tout $1 \leqslant i \leqslant d$,*

$$\lambda_i(L) \geqslant \min_{i \leqslant j \leqslant d} \|b_j^*\|.$$

Remarque 2.14. En particulier, on a $\lambda_1(L) \geqslant \min_{1 \leqslant j \leqslant d} \|b_j^*\|$. Comme $b_1 = b_1^*$, on déduit que si l'on arrive à contrôler les rapports entre les tailles des b_j^*, on va obtenir une relation non triviale entre $\lambda_1(L)$ et $\|b_1\|$.

Nous définissons maintenant deux indicateurs de la qualité d'une base.

2.3.2 Défaut de longueur, défaut d'orthogonalité

Définition 2.15. *Soit L un réseau de base $b = (b_k)_{1 \leqslant k \leqslant d}$ avec $\|b_1\| \leqslant \|b_2\| \leqslant \ldots \leqslant \|b_d\|$. Le i-ème défaut de longueur de b est*

$$\mu_i(b) = \frac{\|b_i\|}{\lambda_i(L)} \geqslant 1.$$

Définition 2.16. *Le défaut d'orthogonalité d'une base $(b_1, \ldots, b_d)$ est*

$$\rho(b) = \prod_{i=1}^{d} \frac{\|b_i\|}{\|b_i^*\|} = \frac{\prod_{i=1}^{d} \|b_i\|}{\text{dét}\,(L)}.$$

Une base b est orthogonale ssi $\rho(b) = 1$.
Le second théorème de Minkowski entraîne

Proposition 2.17. *On a*

$$1 \leqslant \frac{\rho(b)}{\prod_{i=1}^{d} \mu_i(b)} \leqslant d^{d/2}.$$

Remarque 2.18. On voit encore que les notions « base formée de vecteurs courts » et « base presque orthogonale » sont voisines.

2.4 Diverses notions de réduction

La réduction faible est une notion que l'on trouve déjà dans des travaux de Hermite [90] sur les formes quadratiques et consiste, simplement, à « redresser » chaque vecteur de la base en le rapprochant le plus possible de son vecteur de Gram-Schmidt associé.

Définition 2.19. *Une base $(b_i)_{1\leqslant i\leqslant d}$ d'un réseau L est dite* faiblement réduite *(ou* propre*) si les coefficients m_{ij} $(1 \leqslant i \leqslant d, 1 \leqslant j \leqslant i-1)$ du procédé de Gram-Schmidt vérifient $|m_{ij}| \leqslant \frac{1}{2}$.*

Une base quelconque peut aisément être rendue propre, cf. l'algorithme 2.

Algorithme 2 Réduction faible (ou proprification)

Entrée : Une base (b_i) d'un réseau L
Sortie : Une base (b_i) de L faiblement réduite

```
1: pour k = 2 à d faire
2:    pour j = k − 1 à 1 faire
3:       b_k ← b_k − ⌊(b_k|b*_j)/‖b*_j‖²⌉ b_j
4:    fin pour
5: fin pour
```

Remarque 2.20. Le réseau L et la base (b_i^*) sont inchangés.

La réduction suivante est très forte :

Définition 2.21. *Une base $(b_i)_{1\leqslant i\leqslant d}$ d'un réseau L est dite* réduite au sens de Hermite-Korkine-Zolotarev (HKZ) *si*

— *$(b_i)_{1\leqslant i\leqslant d}$ est propre ;*
— *$\forall i = 1, \ldots, d$, b_i^* est un élément de longueur minimale de $\mathrm{proj}_{\mathrm{Vect}_{\mathbb{R}}(b_1,\ldots,b_{i-1})^{\perp}} \mathrm{Vect}_{\mathbb{R}}(b_1, \ldots, b_i)$.*

Elle a le mérite [118] de fournir une très bonne approximation des minima successifs :

Théorème 2.22. *Soit $(b_i)_{1\leqslant i\leqslant d}$ une base HKZ-réduite d'un réseau L, on a pour tout i, $1 \leqslant i \leqslant d$,*

$$\frac{4}{i+3} \leqslant \left(\frac{\|b_i\|}{\lambda_i(L)}\right)^2 \leqslant \frac{i+3}{4},$$

ainsi que de se comporter très bien vis-à-vis du défaut d'orthogonalité.

Remarque 2.23. $\|b_1\|$ est le premier minimum du réseau.

Pour terminer, nous présentons une réduction moins contraignante mais qui, comme nous allons le voir dans la suite, offre un compromis, formidablement efficace et fécond, entre qualité de la base et difficulté de son calcul.

Définition 2.24. *Soit $\delta \in]1/4, 1]$. Une base $(b_1, \ldots, b_d)$ de L est dite LLL-réduite à un facteur δ si*

1. *elle est faiblement réduite ;*
2. *pour tout $1 \leqslant i < d$, $\|b_{i+1}^* + m_{i+1,i} b_i^*\|^2 \geqslant \delta \|b_i^*\|^2$ (condition de Lovász).*

Remarque 2.25. Initialement [129], $\delta = 3/4$.

La condition 2 est équivalente à $\|b_{i+1}^*\|^2 \geqslant (\delta - m_{i+1,i}^2)\|b_i^*\|^2$, ce qui entraîne $\|b_{i+1}^*\|^2 \geqslant (\delta - 1/4)\|b_i^*\|^2$. Cela nous place dans les conditions de la remarque 2.14 et nous obtenons donc un contrôle assez bon des défauts de longueur et d'orthogonalité :

Théorème 2.26. *Soit $\delta \in]1/4, 1]$, on pose $\alpha = 1/(\delta - 1/4)$. Soit $(b_1, \ldots, b_d)$ une base LLL-réduite à un facteur δ de L. Alors*

1. *$\|b_1\| \leqslant \alpha^{(d-1)/4} (\text{dét } L)^{1/d}$;*
2. *Pour tout $1 \leqslant i \leqslant d$, $\|b_i\| \leqslant \alpha^{(d-1)/2} \lambda_i(L)$;*
3. *$\text{dét } L \leqslant \|b_1\| \cdots \|b_d\| \leqslant \alpha^{d(d-1)/4} \text{dét } L$.*

Remarque 2.27. La valeur initiale $\delta = 3/4$ donne $\alpha = 2$. C'est avec cette valeur que l'énoncé précédent est souvent présenté dans la littérature.

2.5 Problèmes algorithmiques

Dans cette section, $\|\cdot\|$ est une norme quelconque sur $\mathbb{R}^n$.

Parmi les problèmes fondamentaux de l'algorithmique des réseaux euclidiens, il y en a deux qui vont jouer un rôle important dans ce manuscrit, sous une forme exacte ou approchée : SVP (pour Shortest Vector Problem) et CVP (pour Closest Vector Problem).

2.5.1 Problème du plus court vecteur non nul

Problème 2.28. (SVP) *Étant donnée une base d'un réseau rationnel L, trouver $u \in L$ qui réalise le premier minimum de L. Problème d'approximation associé : trouver $v \in L \setminus \{0\}$ tel que $\|v\| \leqslant k\lambda_1(L)$ où $k \in \mathbb{R}$ fixé.*

SVP_∞ (SVP pour $\|\cdot\|_\infty$) est NP-difficile [211]. SVP_2 (SVP pour $\|\cdot\|_2$) est NP-difficile pour des réductions polynomiales probabilistes [2], et reste NP-difficile si on tolère un facteur d'approximation $< \sqrt{2}$ [149].

Goldreich et Goldwasser [74] ont démontré qu'approcher SVP à un facteur $\sqrt{d/O(\log d)}$ n'est pas NP-dur.

À ce jour, il n'existe pas d'algorithme polynomial connu pour approcher SVP, dans quelque norme $\ell_p, 1 \leqslant p \leqslant \infty$, que ce soit, à un facteur $f(d)$ où f est un polynôme.

2.5.2 Problème du vecteur le plus proche

Problème 2.29. (CVP) *Étant donnés une base d'un réseau rationnel L et $x \in \mathbb{Q}^n$ trouver $y \in L$ tel que $\|x - y\| = \mathrm{dist}(x, L)$. Problème d'approximation associé : trouver $y \in L \setminus \{0\}$ tel que $\|x - y\| \leqslant k\,\mathrm{dist}(x, L)$ où $k \in \mathbb{R}$ fixé.*

Van Emde Boas [211] a prouvé que CVP est NP-dur pour toute norme $\ell_p, p \geqslant 1$, et ce résultat a été étendu par Dinur, Kindler, Raz et Safra [53] à de petits facteurs d'approximation pour toute norme $\ell_p, 1 \leqslant p < \infty$.

Goldreich, Micciancio, Safra et Seifert [75] ont démontré que CVP ne peut pas être plus facile que SVP.

Goldreich et Goldwasser [74] ont démontré (leur preuve n'est pas constructive toutefois) qu'approcher CVP_2 (CVP pour $\|\cdot\|_2$) à un facteur $\sqrt{d/O(\log d)}$ n'est pas NP-dur, et ils obtiennent le même résultat pour CVP_∞ (CVP pour $\|\cdot\|_\infty$) avec un facteur d'approximation $d/\log d$.

À ce jour, on ne connaît aucun algorithme polynomial pour résoudre CVP, dans quelque norme $\ell_p, 1 \leqslant p \leqslant \infty$, que ce soit, avec un facteur d'approximation polynomial.

2.6 Algorithmes de réduction de réseaux

Nous avons déjà noté qu'un réseau de dimension $d \geqslant 2$ a une infinité de bases et que parmi ces bases, celles composées de vecteurs courts et les plus orthogonaux possibles sont souhaitables. Les algorithmes de réduction de réseau ont pour objectif de nous fournir une de ces bonnes bases à partir d'une base quelconque. Ils nous permettent de résoudre SVP de manière exacte ou approchée et constituent une étape cruciale de pré-calcul pour la résolution exacte ou approchée de CVP.

2.6.1 L'algorithme de Lenstra, Lenstra et Lovász

En 1982, Lenstra, Lenstra et Lovász [129] ont publié l'algorithme 3. Son importance et son utilité sont toujours immenses aujourd'hui.

On constate que la base obtenue est bien réduite au sens de Lovász si l'algorithme termine. Le problème est d'être bien assuré que c'est le cas : le processus d'incrémentation-décrémentation pourrait être infini ! En fait, l'algorithme termine bien et cela en temps polynomial.

Algorithme 3 Algorithme LLL

Entrée : Une base $(b_i)_{1\leqslant i\leqslant d}$ d'un réseau L, un paramètre $\delta \in]1/4, 1]$
Sortie : Une base $(b_i)_{1\leqslant i\leqslant d}$ $\delta-$LLL-réduite de L

1: $k \leftarrow 1$
2: **tant que** $1 \leqslant k \leqslant d$ **faire**
3: Réduire faiblement $(b_1, \ldots, b_k, b_{k+1})$
4: **si** $\|b^*_{k+1}\|^2 \geqslant (\delta - m^2_{k+1,k})\|b^*_k\|^2$ **alors**
5: $k \leftarrow k + 1$
6: **sinon**
7: Échanger b_k et b_{k+1}
8: $k \leftarrow \max(2, k - 1)$
9: **fin si**
10: $j \leftarrow j - 1$
11: **fin du tant que**

Théorème 2.30. *Soit L un réseau de rang d dans $\mathbb{Q}^n$. Si $\delta \in]1/4, 1[$, LLL termine en temps polynomial, au plus $O(d^5 n \ln^3 B)$ opérations avec $B \geqslant \|b_i\|^2$ pour tout i.*

On sait donc obtenir en temps polynomial une base formée de vecteurs « assez » courts, et en particulier, une approximation de SVP à un facteur exponentiel $2^{(d-1)/2}$ près.

Remarque 2.31. Le paramètre δ est classiquement choisi égal à $3/4$. Plus δ est proche de 1, meilleure est – en principe – la qualité de la base réduite, mais plus le processus est lent. L'idéal, en théorie, serait de prendre $\delta = 1$; on ne sait alors plus garantir que l'algorithme termine en temps polynomial. On peut envisager par exemple de faire varier la valeur de δ en cours d'exécution, en l'approchant de 1.

2.6.2 Les algorithmes BKZ et HKZ

Il n'existe pas à ce jour d'algorithmes polynomiaux pour ces deux notions de réduction et il est peu probable que ce soit le cas étant donné le caractère NP-difficile de SVP. On peut obtenir une base HKZ-réduite à l'aide de l'algorithme de Kannan-Fincke-Pohst [102, 68] en un temps $2^{O(d)}$.

Schnorr [187] a introduit une hiérarchie de réductions intermédiaires permettant de parcourir le spectre de la réduction LLL à la réduction HKZ et a défini la famille d'algorithmes correspondants, nommés BKZ (pour Blockwise Korkine-Zolotarev). On peut consulter [85] pour une analyse de ces algorithmes. Ces algorithmes dépendent d'un paramètre $\beta \in [2, d]$, la taille de bloc. Pour $\beta = 2$, on retrouve LLL et pour $\beta = d$, BKZ est identique à HKZ. BKZ retourne une base $(b_i)_{1\leqslant i\leqslant d}$ telle que $\|b_1\| \leqslant \beta^{(d-1)/(\beta-1)}(\text{dét}\, L)^{1/d}$ en un temps $P(d, B)2^{O(\beta)}$, où $P \in \mathbb{Z}[X, Y]$ et B est un majorant de la taille des vecteurs d'entrée.

2.6.3 Implémentations

L'algorithme LLL historique [129] manque d'efficacité pratique. L'utilisation d'arithmétique exacte lors de l'orthogonalisation de Gram-Schmidt est en cause et on a très tôt essayé de la remplacer par de l'arithmétique virgule flottante. Notre lectrice trouvera une excellente présentation de ce sujet dans [164, Chap. 5].

À ce jour, la solution la plus satisfaisante est donnée par les travaux [161, 163, 154]. Ils présentent une variante de LLL, nommée L^2, qui utilise l'arithmétique virgule flottante et retourne de manière prouvée une base LLL-réduite en temps polynomial $O(d^2 n(d + \log B) \log B \cdot \mathcal{M}(d))$, où B est un majorant de la taille des vecteurs d'entrée et $\mathcal{M}(d)$ la complexité du produit de deux entiers de taille d bits.

Dans toutes nos implémentations, nous utilisons la bibliothèque `fplll`[1]. Elle nous fournit des implémentations de référence des algorithmes LLL, BKZ et HKZ.

1. https://github.com/fplll

2.7 Algorithmes de résolution de CVP

Nous n'aurons besoin en pratique que d'algorithmes résolvant des approximations de CVP. La lectrice intéressée par des résolutions exactes pourra consulter [84, 1] par exemple.

2.7.1 L'algorithme du plan le plus proche de Babai

En s'appuyant sur l'algorithme LLL, Babai [7] a proposé l'algorithme 4 : il résout CVP_2, en temps polynomial, avec un facteur d'approximation exponentiel. Pour $x \in \mathbb{R}$, on note $\lfloor x \rceil$ l'entier le plus proche de x, défini par $\lfloor x \rceil = \lfloor x + 1/2 \rfloor$. Nous l'utiliserons pour la synthèse de routines d'évaluation de fonctions dans la section 4.5.

Algorithme 4 Algorithme du plan le plus proche de Babai

Entrée : Une base LLL-réduite $(b_i)_{1 \leqslant i \leqslant d}$, son orthogonalisée de Gram-Schmidt $(b_i^*)_{1 \leqslant i \leqslant d}$, un vecteur v
Sortie : Une approximation à un facteur $2^{d/2}$ d'une solution au CVP_2 de v

1: $t \leftarrow v$
2: $j \leftarrow d$
3: **tant que** $j \geqslant 1$ **faire**
4: $\quad t \leftarrow t - \left\lfloor \frac{\langle t, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \right\rceil b_j$
5: $\quad j \leftarrow j - 1$
6: **fin du tant que**
7: Renvoyer $v - t$

2.7.2 L'algorithme du plongement de Kannan

En travaillant sur le problème de synthèse de filtre RIF présenté dans la section 4.6, il nous est apparu que l'algorithme 4, reposant sur LLL, ne retournait pas dans ce cadre des résultats aussi bons que pour l'évaluation de fonction. Nous avons donc été amenés à considérer plutôt un autre algorithme de résolution approchée de CVP_2, celui du plongement de Kannan [104] en nous autorisant à le coupler à trois algorithmes de réduction de réseau, à savoir LLL, BKZ et HKZ.

Nous utilisons la technique du plongement de Kannan de la façon suivante : étant donnés une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ de L et un vecteur cible $\mathbf{t}$, nous formons le réseau L' de dimension $(d+1)$ de $\mathbb{R}^{\ell+1}$ engendré par les vecteurs $\mathbf{b}'_i := \begin{pmatrix} \mathbf{b}_i \\ 0 \end{pmatrix}$ et $\mathbf{t}' := \begin{pmatrix} \mathbf{t} \\ \gamma \end{pmatrix}$, où γ est un paramètre réel que l'on choisit ultérieurement. Si $\mathbf{u}$ est un vecteur court du réseau L' (cela peut être un vecteur d'une base réduite de L'), il existe $u_1, \ldots, u_d, u_{d+1}$ tels que $\mathbf{u} = \sum_{k=1}^{d} u_k \mathbf{b}'_k + u_{d+1}\mathbf{t}'$; d'où

$$\|\mathbf{u}\|_2^2 = \left\| \sum_{k=1}^{d} u_k \mathbf{b}_k + u_{d+1}\mathbf{t} \right\|_2^2 + \gamma^2 u_{d+1}^2.$$

Comme $\mathbf{u}$ est supposé court, $\| \sum_{k=1}^{d} u_k \mathbf{b}_k + u_{d+1}\mathbf{t} \|_2^2$ est petit. Si $u_{d+1} = \pm 1$, $\mp \sum_{k=1}^{d} u_k \mathbf{b}_k \in L$ est proche du vecteur $\mathbf{t}$. Autrement dit, si nous sommes capables de trouver une base de L' formée de vecteurs courts, nous pouvons espérer obtenir un vecteur de L très proche de t.

Au cours de nos expérimentations, le choix (heuristique) $\gamma = \max_{k=1}^{d} \|\mathbf{b}_k\|_2$ a donné un vecteur avec $u_{d+1} = \pm 1$ dans tous les cas (voir [66, Ch. 4.3.4] pour une justification théorique).

Remarque 2.32. Usuellement [104, 103, 54] l'idée de Kannan est utilisé de manière plus fine, avec plusieurs appels à la réduction de réseau, ce qui permet d'obtenir de bonnes garanties théoriques de qualité. La version simplifiée que nous présentons, cf. algorithme 5, est un compromis qui nous a permis d'avoir un algorithme efficace qui nous renvoie une solution de bonne qualité.

Algorithme 5 Plongement de Kannan

Entrée : Une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ de $L \subset \mathbb{R}^\ell$, un vecteur cible $\mathbf{t} \in \mathbb{R}^\ell$, un algorithme de réduction de réseau $LBR \in \{LLL, BKZ, HKZ\}$

Sortie : Un d-uplet $\mathbf{m} = (m_1, \cdots, m_d) \in \mathbb{Z}^d$ tel que $\sum_{k=1}^{d} m_k \mathbf{b}_k$ soit une solution approchée du CVP_2 de $\mathbf{t}$, matrice de changement de base $\mathbf{U} \in \mathbb{Z}^{d \times d}$

1: $\gamma \leftarrow \max_{1 \leqslant k \leqslant d} \|\mathbf{b}_k\|_2$
// Construire la base
2: $\mathbf{B}' \leftarrow \begin{pmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_d & \mathbf{t} \\ 0 & 0 & \cdots & 0 & \gamma \end{pmatrix}$
// Effectuer la réduction de réseau (LLL, BKZ ou HKZ) sur $\mathbf{B}'$
// $\mathbf{C}' = \left(c'_{i,j}\right) \in \mathbb{R}^{(\ell+1)\times(d+1)}$ est la base réduite
// $\mathbf{U}' = \left(u'_{i,j}\right) \in \mathbb{Z}^{(d+1)\times(d+1)}$ est la matrice de
// changement de base (c.-à-d. $\mathbf{C}' = \mathbf{B}'\mathbf{U}'$)
3: $(\mathbf{C}', \mathbf{U}') \leftarrow$ LatticeReduce($\mathbf{B}'$)
// Récupérer la matrice de changement de base pour $\mathbf{B}$
// (c.-à-d. d'abord d lignes et colonnes de $\mathbf{U}'$)
4: $\mathbf{U} \leftarrow \left(u'_{i,j}\right)_{1 \leqslant i \leqslant d, 1 \leqslant j \leqslant d}$
// Extraire l'élément situé sur les dernières ligne et colonne de $\mathbf{C}'$
5: $\gamma' \leftarrow c'_{\ell+1,d+1}$
// Construire la sortie à l'aide de γ' et
// de la dernière colonne de $\mathbf{U}'$
6: $s \leftarrow 1$
7: **si** $\gamma' = -\gamma$ **alors**
8: $\quad s \leftarrow -1$
9: **fin si**
10: **pour** $i = 1$ **à** d **faire**
11: $\quad m_i \leftarrow s \cdot u'_{i,d+1}$
12: **fin pour**

Chapitre 3

Quelques rappels de théorie de l'approximation

Dans ce chapitre, on regroupe des énoncés théoriques et algorithmiques, sans preuve, relatifs à l'approximation polynomiale. Nous travaillerons principalement avec des fonctions continues à valeurs réelles sur un intervalle compact $[a, b]$, $a, b \in \mathbb{R}$, $a \leqslant b$. Nous noterons $\mathcal{C}([a, b])$ l'espace vectoriel sur $\mathbb{R}$ des fonctions continues sur $[a, b]$. Les références [25, 31, 49, 69, 146, 175, 204, 181, 186] sont d'excellentes présentations des domaines évoqués brièvement ici.

3.1 Approximations polynomiales

Dans le cadre de l'évaluation de fonction, on travaille usuellement avec l'une de ces deux normes :

— la norme aux moindres carrés L^2 : étant donnée une fonction poids w [1], si $\mathrm{d}x$ désigne la mesure de Lebesgue, on dira que
$$g \in L^2([a, b], w, \mathrm{d}x)$$
si
$$\int_a^b w(x)|g(x)|^2 \mathrm{d}x < \infty,$$
et on définit alors
$$\|g\|_{2,w} = \sqrt{\int_a^b w(x)|g(x)|^2 \mathrm{d}x};$$

— la norme sup (connue aussi sous le nom de norme de Tchebychev, norme infini, norme L^∞) : si g est bornée sur $[a, b]$, on pose
$$\|g\|_\infty = \sup_{x \in [a,b]} |g(x)|,$$
(pour une fonction continue g, on a $\|g\|_\infty = \max_{x \in [a,b]} |g(x)|$).

Pour ces deux normes se pose la question fondamentale suivante :

Problème 3.1. *(Meilleure approximation) Étant donnés $f \in \mathcal{C}([a, b])$ et $n \in \mathbb{N}$, minimiser $\|f - p\|$ où p parcourt l'espace $\mathbb{R}_n[x]$ des polynômes à coefficients réels et de degré au plus n.*

Dans le cas L^2, la réponse est aisée. L'espace $\mathcal{C}([a, b])$ est un sous-ensemble de l'espace de Hilbert $L^2([a, b], w, \mathrm{d}x)$. Il y a existence et unicité de la meilleure approximation polynomiale de degré au plus n : il s'agit de la projection $p = \mathrm{pr}_\perp(f)$ de f sur $\mathbb{R}_n[x]$. Nous reviendrons sur le cas L^2 lorsque nous parlerons de polynômes orthogonaux, dans la section 3.3. La situation dans le cas L^∞ est plus complexe, comme nous allons le voir maintenant.

1. On convient ici que cela signifie que $w \in \mathcal{C}(]a, b[)$ et à valeurs strictement positives presque partout.

3.1.1 Densité des polynômes dans $(\mathcal{C}([a,b]), \|.\|_\infty)$

Pour toute $f \in \mathcal{C}([a,b])$ et $n \in \mathbb{N}$, on pose

$$E_n(f) = \inf_{p \in \mathbb{R}_n[x]} \|f - p\|_\infty.$$

Rappelons tout d'abord que $E_n(f) \to 0$ quand $n \to \infty$:

Théorème 3.2 (Weierstraß, 1885 [214]). *Pour tout $f \in \mathcal{C}([a,b])$ et pour tout $\varepsilon > 0$, il existe $n \in \mathbb{N}$, $p \in \mathbb{R}_n[x]$ tel que $\|p - f\|_\infty < \varepsilon$.*

De nombreuses preuves de ce résultat ont été publiées : pour n'en citer que quelques-unes, il y a celles de Runge (1885), Picard (1891), Lerch (1892 et 1903), Volterra (1897) Lebesgue (1898), Mittag-Leffler (1900), Fejér (1900 et 1916), Landau (1908), la Vallée Poussin (1908), Jackson (1911), Sierpinski (1911), Bernstein (1912), Montel (1918). Le texte [172] parle de manière intéressante de la contribution de Weierstraß à la théorie de l'approximation et en particulier du théorème 3.2.

Dans sa preuve, Bernstein a utilisé la famille de polynômes (appelés à présent polynômes de Bernstein)

$$B_n(g,x) = \sum_{k=0}^{n} \binom{n}{k} g(k/n) x^k (1-x)^{n-k} \text{ pour } g \in \mathcal{C}([0,1]).$$

et c'est un des aspects remarquables de sa preuve que de nous fournir une suite explicite de polynômes qui converge vers f. Nous disposons même d'une vitesse de convergence.

Définition 3.3. *Le module de continuité de f est la fonction ω définie par*

$$\text{pour tout } \delta > 0, \qquad \omega(\delta) = \sup_{\substack{|x-y| < \delta, \\ x,y \in [a,b]}} |f(x) - f(y)|.$$

Proposition 3.4. *Soit f une fonction continue sur $[0,1]$, ω son module de continuité, alors*

$$\|f - B_n(f,x)\|_\infty \leqslant \tfrac{9}{4}\omega\left(n^{-\frac{1}{2}}\right).$$

Remarque 3.5. Comme nous le verrons dans le théorème 3.42, on dispose de familles d'approximants polynomiaux qui, suivant la régularité de f, peuvent converger bien plus rapidement.

Corollaire 3.6. *Lorsque f est Lipschitz continue*[2]*, on a $E_n(f) = O(n^{-1/2})$.*

Les polynômes de Bernstein sont très utilisés aujourd'hui, en infographie ou en optimisation globale par exemple [64].

3.1.2 Approximation minimax

La proposition suivante nous dit en particulier que l'infimum $E_n(f)$ est atteint :

Proposition 3.7. *Soit $(E, \|\cdot\|)$ un $\mathbb{R}$-espace vectoriel normé, soit F un sous-espace de dimension finie de $(E, \|\cdot\|)$. Pour tout $f \in E$, il existe $p \in F$ tel que $\|p - f\| = \min_{q \in F} \|q - f\|$. De plus, l'ensemble des meilleures approximations de $f \in E$ est convexe.*

Cette proposition nous permet d'introduire la terminologie suivante : on appelle *approximation minimax* une meilleure approximation au sens L^∞. Cela correspond au fait que, pour $f \in \mathcal{C}([a,b])$, on cherche une approximation $p \in \mathbb{R}_n[X]$ telle que $\|f - p\|_\infty$ minimise $\max_{x \in [a,b]} |f(x) - q(x)|$, où q parcourt $\mathbb{R}_n[X]$.

La meilleure approximation L^2 est unique, ce qui n'est pas toujours le cas dans le cadre L^∞.

Dans le cas L^∞, il est nécessaire d'introduire une condition supplémentaire que l'on appelle condition de Haar.

2. Cela signifie qu'il existe $K \geqslant 0$ telle que $|f(x) - f(y)| \leqslant K|x - y|$ pour tous $x, y \in [0,1]$

Définition 3.8. *On considère $n+1$ fonctions $\varphi_0, \dots, \varphi_n$ définies sur $[a,b]$. On dit que $\varphi_0, \dots, \varphi_n$ satisfont à la condition de Haar si*

a) les φ_i sont continues ;

b) et une des assertions équivalentes suivantes est vérifiée :
- *pour tous $x_0, x_1, \dots, x_n \in [a,b]$,*

$$\begin{vmatrix} \varphi_0(x_0) & \cdots & \varphi_n(x_0) \\ \vdots & & \vdots \\ \varphi_0(x_n) & \cdots & \varphi_n(x_n) \end{vmatrix} = 0 \quad \Leftrightarrow \quad \exists i \neq j, x_i = x_j;$$

- *étant donnés des éléments distincts $x_0, \dots, x_n \in [a,b]$ et des valeurs $y_0, \dots, y_n$, il existe un unique interpolant*

$$p = \sum_{k=0}^{n} \alpha_k \varphi_k, \text{ avec } \alpha_k \in \mathbb{R}, \forall k = 0, \dots, n,$$

  tel que $p(x_i) = y_i, \forall i = 0, \dots, n$;
- *les $\varphi_k, k = 0, \dots, n$, sont $\mathbb{R}$-linéairement indépendants et tout $p = \sum_{k=0}^{n} \alpha_k \varphi_k \neq 0$ a au plus n zéros distincts dans $[a,b]$.*

Un ensemble de fonctions qui satisfait à la condition de Haar est appelé système de Tchebychev. Un exemple caractéristique est donné par $\varphi_i(x) = x^i$, pour lequel nous avons un déterminant de Vandermonde :

$$\begin{vmatrix} \varphi_0(x_0) & \cdots & \varphi_n(x_0) \\ \vdots & & \vdots \\ \varphi_0(x_n) & \cdots & \varphi_n(x_n) \end{vmatrix} = \begin{vmatrix} 1 & \cdots & x_0^n \\ \vdots & & \vdots \\ 1 & \cdots & x_n^n \end{vmatrix} = \prod_{0 \leqslant i < j \leqslant n} (x_j - x_i).$$

Remarque 3.9. Les familles de fonctions suivantes sont des systèmes de Tchebychev :
- $\{e^{\lambda_0 x}, \dots, e^{\lambda_n x}\}$ pour $\lambda_0 < \lambda_1 < \cdots < \lambda_n$;
- $\{1, \cos x, \sin x, \dots, \cos(nx), \sin(nx)\}$ sur $[a,b]$ où $0 \leqslant a < b < 2\pi$;
- $\{x^{\alpha_0}, \dots, x^{\alpha_n}\}$, $\alpha_0 < \cdots < \alpha_n$, sur $[a,b]$ avec $a > 0$.

Si $\{\varphi_0, \dots, \varphi_n\}$ est un système de Tchebychev sur $[a,b]$, tout élément de $\mathrm{Vect}_{\mathbb{R}}\{\varphi_0, \dots, \varphi_n\}$ est appelé polynôme généralisé.

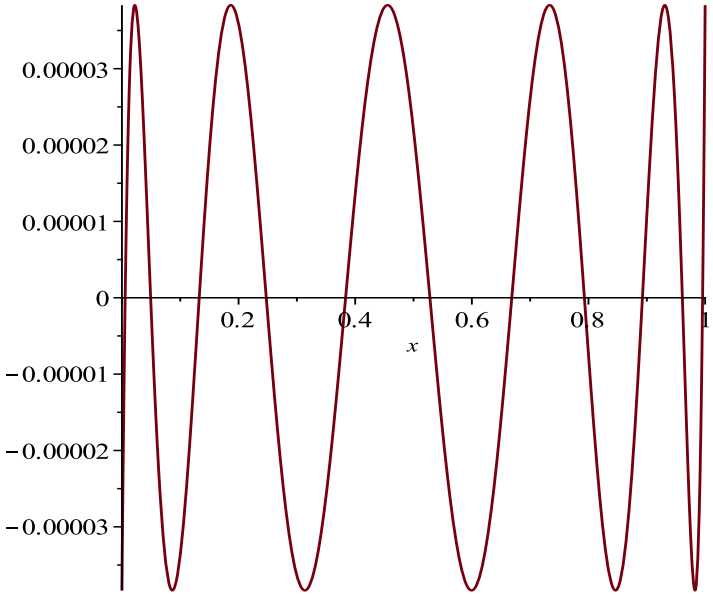
Outre sa beauté, la caractérisation suivante de l'approximation minimax, souvent attribuée à Tchebychev ou à Borel, a été cruciale pour l'élaboration d'un procédé effectif de calcul de ce minimax.

Théorème 3.10. *[Théorème d'alternance. Kirchberger (1902)] Soit $\{\varphi_0, \dots, \varphi_n\}$ un système de Tchebychev sur $[a,b]$. Soit $f \in \mathcal{C}([a,b])$. Un polynôme généralisé $p = \sum_{k=0}^{n} \alpha_k \varphi_k$ est la meilleure approximation, au sens L^∞, (ou l'approximation minimax) de f ssi il existe $n+2$ points $x_0, \dots, x_{n+1}$, $a \leqslant x_0 < x_1 < \cdots < x_{n+1} \leqslant b$ tel que, pour tout k,*

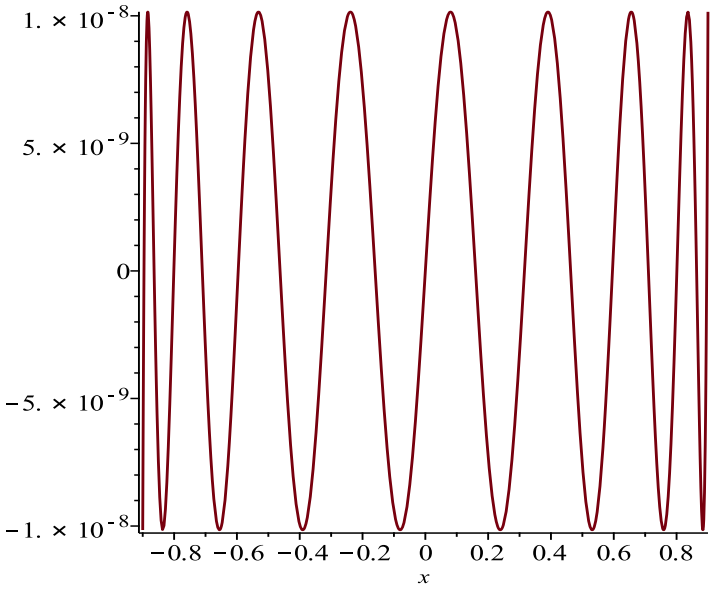
$$f(x_k) - p(x_k) = (-1)^k (f(x_0) - p(x_0)) = \pm \|f - p\|_\infty.$$

Autrement dit, p est la meilleure approximation si et seulement si la fonction d'erreur $f - p$ a, au moins, $n+2$ extrema, tous globaux (de même valeur absolue) et avec des signes qui alternent.

Exemple 3.11. Soit $f : x \in [0,1] \longmapsto e^{1/\cos(x)}, p^* = \sum_{k=0}^{10} c_k x^k$ son approximation minimax. Le graphe de la fonction d'erreur $\varepsilon = f - p$ est :



Exemple 3.12. Soit $f : x \in [-0.9, 0.9] \longmapsto \arctan(x), p^* = \sum_{k=0}^{15} c_k x^k$ son approximation minimax. Le graphe de la fonction d'erreur $\varepsilon = f - p$ est :



Exemple 3.13. La meilleure approximation de $\cos$ sur $[0, 10\pi]$ sur le système de Tchebychev $\{1, x, x^2\}$ est la fonction constante 0 ! Cela reste vrai pour $\{1, x, \ldots, x^h\}$ tant que $h \leqslant 9$.

Le résultat que nous présentons maintenant est aussi un élément clé pour l'élaboration d'un algorithme de calcul du minimax.

Théorème 3.14. *(La Vallée Poussin) Soit $f \in \mathcal{C}([a, b])$. Soit $\{\varphi_0, \ldots, \varphi_n\}$ un système de Tchebychev sur $[a, b]$, soit $p \in \mathrm{Vect}_{\mathbb{R}}\{\varphi_0, \ldots, \varphi_n\}$. S'il existe $x_0 < x_1 < \cdots < x_{n+1}$ tel le signe de $p - f$ alterne aux points x_i, alors*

$$\min_i |f(x_i) - p(x_i)| \leqslant E_n(f) \leqslant \|f - p\|_\infty,$$

où $E_n(f) = \min_{q \in \mathrm{Vect}_{\mathbb{R}}\{\varphi_i\}} \|f - q\|_\infty$.

Remarque 3.15. Les énoncés des théorèmes 3.10 and 3.14 restent valables si l'on remplace $[a, b]$ par un sous-ensemble compact de $\mathbb{R}$ contenant au moins $n + 2$ points.

Remez [179] a publié en 1934 l'algorithme 6 pour approcher, aussi finement que désiré, le polynôme minimax. Cet algorithme est utilisé pour la synthèse de fonctions mathématiques mais c'est sa variante, due à Parks et McClellan [169], dans le cadre de la synthèse de filtres en traitement du signal qui a connu un immense succès.

Algorithme 6 Second algorithme de Remez

Entrée : Un intervalle $[a, b]$, une fonction $f \in \mathcal{C}([a, b])$, un entier n, un système de Tchebychev $\{\varphi_k\}_{0 \leqslant k \leqslant n}$, une tolérance Δ.

Sortie : Une approximation du polynôme minimax de degré n de f sur le système $\{\varphi_k\}$.

1: Choisir $n + 2$ points $x_0 < x_1 < \cdots < x_{n+1}$ dans $[a, b]$, $\delta \leftarrow 1, \varepsilon \leftarrow 0$.
2: **tant que** $\delta \geqslant \Delta|\varepsilon|$ **faire**
3: Déterminer les solutions $a_0, \ldots, a_n$ et ε du système linéaire

$$\sum_{k=0}^{n} a_k \varphi_k(x_j) - f(x_j) = (-1)^j \varepsilon,\ j = 0, \ldots, n + 1.$$

4: Choisir $x_{\text{new}} \in [a, b]$ tel que

$$\|p - f\|_\infty = |p(x_{\text{new}}) - f(x_{\text{new}})|, \text{avec}\, p = \sum_{k=0}^{n} a_k \varphi_k.$$

5: Remplacer un des x_i par x_{new}, de telle sorte que le signe de $p - f$ alterne aux points de la nouvelle discrétisation $x_{0,\text{new}}, \ldots, x_{n+1,\text{new}}$.
6: $\delta \leftarrow |p(x_{\text{new}}) - f(x_{\text{new}})| - |\varepsilon|$.
7: **fin du tant que**
8: Renvoyer p.

Pour plus de détails sur cet algorithme, on peut consulter [31, 175]. On y trouvera notamment ce résultat sur la vitesse de convergence de l'algorithme de Remez.

Théorème 3.16. *Soit p_k la valeur de p après $k(n+2)$ boucles, et p^* l'approximation minimax. Il existe $\theta \in]0, 1[$ tel que $\|p_k - p^*\|_\infty = O(\theta^k)$.*

Si l'on ajoute certaines conditions (peu restrictives), on peut remplacer $\mathrm{O}(\theta^k)$ par $\mathrm{O}(\theta^{2^k})$ [212].

3.2 Interpolation et approximation

Nous restreignons de nouveau notre exposé aux polynômes de $\mathbb{R}_n[x]$. L'interpolation polynomiale est un outil indispensable lorsque l'on ne dispose que de très peu de données sur un phénomène mais, comme nous allons maintenant le voir, elle peut aussi constituer un outil remarquablement efficace pour bâtir de très bonnes approximations d'une fonction.

On rappelle tout d'abord la formule d'interpolation de Lagrange. Soient $x_0, \ldots, x_n \in [a, b]$ distincts deux à deux. Pour tout $j = 0, \ldots, n$, on note

$$\ell_j(x) = \prod_{\substack{0 \leqslant k \leqslant n, \\ k \neq j}} \frac{x - x_k}{x_j - x_k}.$$

On a donc $\deg \ell_j = n$ et $\ell_j(x_i) = \delta_{i,j}$ pour tous $0 \leqslant i, j \leqslant n$. Les polynômes ℓ_j, $0 \leqslant j \leqslant n$, forment une base de $\mathbb{R}_n[x]$.

Soit p le polynôme d'interpolation défini par $p(x_i) = y_i$ où $y_i \in \mathbb{R}, i = 0, \dots, n$, on peut écrire p sous la forme

$$p(x) = \sum_{i=0}^{n} y_i \ell_i(x).$$

Est-ce que l'interpolation peut être utile pour notre problème de départ, à savoir le problème de l'approximation L^∞ ? En fait, le choix des points d'interpolation joue un rôle clé, comme nous allons le voir.

Commençons par écarter une tentation immédiate : est-ce que plus le nombre de points d'interpolation est grand, meilleure est la qualité d'approximation ? Eh bien, lorsque l'on considère des points équidistants, l'erreur d'approximation peut croître avec le nombre de points, comme le montre l'exemple de Runge [72, Chap. 2]. On considère la fonction $f : x \in [-1, 1] \mapsto \frac{1}{1+5x^2}$ et $p_n \in \mathbb{R}_n[x]$ son polynôme d'interpolation aux points $-1 + \frac{2k}{n}$, $0 \leqslant k \leqslant n$. On a $\lim_{n\to+\infty} \|f - p_n\|_\infty = +\infty$.

En résumé : il vaut mieux ne pas utiliser une famille de points équidistants quand on veut utiliser l'interpolation pour approcher une fonction continue en norme sup. Y a t-il de meilleurs choix ? Le résultat suivant nous dit que non, si l'on suppose seulement la fonction f continue.

Théorème 3.17 (Faber)**.** *Pour tout n, soit un système de familles de nœuds d'interpolation $\xi_0^{(n)}, \dots, \xi_n^{(n)} \in [a, b]$. Il existe une fonction $f \in \mathcal{C}([a,b])$ telle que la suite d'erreurs $(\|f - p_n\|_\infty)_{n\in\mathbb{N}}$ ne soit pas bornée, où $p_n \in \mathbb{R}_n[x]$ désigne le polynôme qui interpole f aux $\xi_0^{(n)}, \dots, \xi_n^{(n)}$.*

Il ne faut pas se décourager. Si l'on exige un peu plus de régularité de f, la situation s'arrange. Commençons par cet analogue de la formule de Taylor-Lagrange.

Théorème 3.18. *Soit $a \leqslant x_0 < \cdots < x_n \leqslant b$, et soit $f \in \mathcal{C}^{n+1}([a, b])$. Soit $p \in \mathbb{R}_n[x]$ tel que $f(x_i) = p(x_i)$ pour tout $i = 0, \dots, n$. Alors, pour tout $x \in [a, b]$, il existe $\xi_x \in (a, b)$ tel que*

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} W(x), \qquad W(x) = \prod_{i=0}^{n} (x - x_i).$$

Ce résultat nous encourage à rechercher des familles de x_i qui rendent $\|W\|_\infty$ aussi petit que souhaité. Il est temps d'introduire les polynômes de Tchebychev.

3.2.1 Polynômes de Tchebychev

On suppose, sans perte de généralité, $[a, b] = [-1, 1]$. Le n-ème polynôme de Tchebychev de première espèce est défini par

$$T_n(\cos t) = \cos(nt), \text{ pour tout } t \in [0, \pi].$$

On peut aussi définir la suite $(T_n)_{n\in\mathbb{N}}$ par

$$T_0(x) = 1, T_1(x) = x, T_{n+2}(x) = 2xT_{n+1}(x) - T_n(x), \forall n \in \mathbb{N}. \tag{3.1}$$

Parmi leurs nombreuses propriétés agréables, il y a le résultat suivant qui va nous suggérer une famille de bons points d'interpolation.

Proposition 3.19. *Soit $n \in \mathbb{N}, n \neq 0$. Le minimum de l'ensemble*

$$\left\{ \max_{x\in[-1,1]} |p(x)| : p \in \mathbb{R}_n[x], p \textit{ unitaire} \right\}$$

est atteint, de manière unique, par $T_n/2^{n-1}$ et vaut par conséquent 2^{-n+1}.

Si l'on pose $W(x) = 2^{-n} T_{n+1}(x)$, cela donne les points d'interpolation

$$\mu_k = \cos\left(\frac{(k + 1/2)\pi}{n+1}\right), k = 0, \dots, n, \tag{3.2}$$

appelés nœuds de Tchebychev de première espèce et $\|W\|_\infty = 2^{-n}$.

Une autre famille importante est constituée des polynômes de Tchebychev de seconde espèce $U_n(x)$, définis par

$$U_n(\cos t) = \frac{\sin((n+1)t)}{\sin(t)}, \text{ pour tout } t \in [0, \pi],$$

ou encore par

$$U_0(x) = 1, U_1(x) = 2x, U_{n+2}(x) = 2xU_{n+1}(x) - U_n(x), \forall n \in \mathbb{N}.$$

Pour tout $n \geqslant 0$, nous avons $\frac{\mathrm{d}}{\mathrm{d}x} T_n = nU_{n-1}$. Les extrema de T_{n+1} sont -1, 1 et les zéros de U_n, c'est-à-dire

$$\nu_k = \cos\left(\frac{k\pi}{n}\right), k = 0, \ldots, n, \tag{3.3}$$

appelés nœuds de Tchebychev de seconde espèce. Si l'on considère $W(x) = 2^{-n+1}(1-x^2)U_{n-1}(x)$, nous avons $\|W\|_\infty = 2^{-n+1}$.

On a $\deg T_n = \deg U_n = n$ pour tout $n \in \mathbb{N}$. Donc, en particulier, la famille $(T_k)_{0 \leqslant k \leqslant n}$ est une base de $\mathbb{R}_n[x]$. Nous concluons cette section en présentant des énoncés qui permettent de calculer rapidement les coefficients des polynômes d'interpolation aux nœuds de Tchebychev (on appellera ces polynômes *interpolants de Tchebychev*), exprimés dans la base $(T_k)_{0 \leqslant k \leqslant n}$.

On désigne par $\sum''$ une somme dont le premier et le dernier termes sont divisés par deux.

Proposition 3.20. *(Orthogonalité discrète) Let $j, \ell \in \{0, \ldots, n\}$.*

i. Nous avons

$$\sum_{k=0}^{n} T_j(\mu_k) T_\ell(\mu_k) = \begin{cases} 0, & j \neq \ell, \\ n+1, & j = \ell = 0, \\ \frac{n+1}{2}, & j = \ell \neq 0. \end{cases}$$

ii. Nous avons

$$\sum_{k=0}^{n}{}'' T_j(\nu_k) T_\ell(\nu_k) = \begin{cases} 0, & j \neq \ell, \\ n, & j = \ell \in \{0, n\}, \\ \frac{n}{2}, & j = \ell \notin \{0, n\}. \end{cases}$$

Cette propriété nous permet d'obtenir des formules explicites de calcul des coefficients des interpolants de Tchebychev ($\sum'$ indique que le premier terme de la somme est divisé par deux).

Proposition 3.21.

i. Si $p_{1,n} = \sum'_{0 \leqslant i \leqslant n} c_{1,i} T_i \in \mathbb{R}_n[x]$ interpole f sur l'ensemble $\{\mu_k, 0 \leqslant k \leqslant n\}$, alors

$$c_{1,i} = \frac{2}{n+1} \sum_{k=0}^{n} f(\mu_k) T_i(\mu_k).$$

ii. De même, si $p_{2,n} = \sum''_{0 \leqslant i \leqslant n} c_{2,i} T_i$ interpole f aux points de l'ensemble $\{\nu_k, 0 \leqslant k \leqslant n\}$, alors

$$c_{2,i} = \frac{2}{n} \sum_{k=0}^{n}{}'' f(\nu_k) T_i(\nu_k).$$

Remarque 3.22. Une conséquence remarquable de cet énoncé est que l'on peut calculer ces coefficients à l'aide de la transformée de Fourier rapide ! On passe donc d'une complexité apparente en $O(n^2)$ à un coût en $O(n \log n)$.

Remarque 3.23. L'efficacité de ce calcul et le fait que l'interpolation aux nœuds de Tchebychev donne lieu à d'excellentes approximations polynomiales sont des points-clés du système de calcul numérique Chebfun [3] mis au point par N. Trefethen et son groupe. Toute fonction y est représentée par un interpolant de Tchebychev, de degré suffisamment élevé pour que l'erreur relative d'approximation (non certifiée !) soit de l'ordre de 2^{-53}, la qualité de calcul associée au format binary64.

3. http://www.chebfun.org/

Lorsqu'on évalue un polynôme dans la base monomiale, la méthode de prédilection est celle du schéma d'évaluation de Horner [156, Algorithme 6.3]. Dans la base de Tchebychev, on dispose d'une méthode analogue due à Clenshaw [146, §2.4.1]. Étant donnés $n+1$ coefficients $c_0, \ldots, c_n$ et un point $t \in [-1,1]$, on souhaite évaluer la somme $\sum_{k=0}^{n} c_k T_k(t)$ plus efficacement que par l'approche naïve en $O(N^2)$. La méthode de Clenshaw s'appuie sur la récurrence (3.1) et elle s'exécute en $O(n)$ opérations arithmétiques.

Algorithme 7 Schéma d'évaluation de Clenshaw

Entrée : Des coefficients $c_0, \ldots, c_n$, un point $t \in [-1,1]$
Sortie : $\sum_{k=0}^{n} c_k T_k(t)$
1: $b_{n+1} \leftarrow 0$, $b_n \leftarrow c_n$
2: **pour** $k = n-1, n-2, \ldots, 1$ **faire**
3: $\quad b_k \leftarrow 2tb_{k+1} - b_{k+2} + c_k$
4: **fin pour**
5: renvoyer $c_0 + tb_1 - b_2$

Remarque 3.24. L'intégration d'un interpolant aux nœuds de Tchebychev est aisée et rapide. Les polynômes de Tchebychev de première espèce satisfont

$$\int_{-1}^{1} T_k(x)\mathrm{d}x = \begin{cases} \frac{2}{1-k^2}, & k \in 2\mathbb{N}, \\ 0, & k \notin 2\mathbb{N}. \end{cases}$$

Donc, si $p = \sum_{k=0}^{n} c_k T_k$ est le polynôme d'interpolation de f aux nœuds de Tchebychev, l'intégrale de f est approchée par

$$\int_{-1}^{1} p(x)\mathrm{d}x = \sum_{\substack{0 \leqslant k \leqslant n \\ k \in 2\mathbb{N}}} \frac{2c_k}{1-k^2}.$$

Comme les coefficients c_k peuvent être calculés en $O(n \log n)$ opérations arithmétiques en utilisant la FFT, cela entraîne une complexité en $O(n \log n)$ pour le calcul de l'intégrale approchée.

3.3 Polynômes orthogonaux - Séries de Tchebychev

Nous allons voir que les polynômes orthogonaux peuvent aussi fournir de très bons approximants polynomiaux.

3.3.1 Polynômes orthogonaux

Soit $]a,b[\subset \mathbb{R}$ un intervalle ouvert (non nécessairement borné), et soit w une fonction poids. On suppose

$$\forall n \in \mathbb{N}, \quad \int_a^b |x|^n w(x)\mathrm{d}x < \infty.$$

C'est le cas, par exemple, si $]a,b[$ est borné et

$$\int_a^b w(x)\mathrm{d}x < \infty.$$

Soit

$$\mathcal{E}(w) = \left\{ f \in \mathcal{C}(]a,b[) : \|f\|_2 := \left(\int_a^b f(x)^2 w(x)\mathrm{d}x \right)^{1/2} < \infty \right\}.$$

Notons que $\mathbb{R}[x] \subset \mathcal{E}(w)$. L'espace $\mathcal{E}(w)$ est muni d'un produit scalaire

$$\langle f, g\rangle = \int_a^b f(x)g(x)w(x)\mathrm{d}x;$$

et $\|\cdot\|_2$ est la norme associée à ce produit scalaire.

Définition 3.25. *Une famille de polynômes orthogonaux associée à w est une suite $(p_n) \in \mathbb{R}[x]^{\mathbb{N}}$ avec $\deg p_k = k$ pour tout k, et*

$$i \neq j \quad \Rightarrow \quad \langle p_i, p_j\rangle = 0.$$

Théorème 3.26. *Pour tout poids w, il existe une famille de polynômes orthogonaux associée à w. Si de plus on exige que les p_k soient tous unitaires, alors cette famille est unique.*

L'énoncé suivant fournit un moyen récursif de calculer une suite de polynômes orthogonaux. Aussi, l'adaptation de la méthode de Clenshaw présentée dans l'algorithme 7 permet, grâce à cet énoncé, d'évaluer en temps linéaire tout polynôme exprimé dans une base de polynômes orthogonaux.

Théorème 3.27. *Les polynômes $(p_n)_{n\in\mathbb{N}}$ satisfont la relation de récurrence*

$$p_n(x) = (x - \alpha_n)p_{n-1}(x) - \beta_n p_{n-2}(x) \qquad (n \geqslant 2)$$

avec

$$\alpha_n = \frac{\langle xp_{n-1}, p_{n-1}\rangle}{\|p_{n-1}\|_2^2}, \qquad \beta_n = \frac{\|p_{n-1}\|_2^2}{\|p_{n-2}\|_2^2}.$$

Exemple 3.28.

$]-1,1[$	$w(x) = (1-x^2)^{-1/2}$	polynômes de Tchebychev de 1ère espèce (à normalisation près)
$]-1,1[$	$w(x) = 1$	polynômes de Legendre
$]0,+\infty[$	$w(x) = e^{-x}$	polynômes de Laguerre
$]-\infty,+\infty[$	$w(x) = e^{-x^2}$	polynômes de Hermite

Théorème 3.29. *Pour tout poids w et pour tout n, le polynôme p_n a n zéros distincts dans $]a,b[$.*

Théorème 3.30. *Soit $f \in \mathcal{E}(w)$, $n \in \mathbb{N}$. Il existe une unique meilleure approximation, au sens $L_2(w)$, de f dans $\mathbb{R}_n[x]$, notée $p_{2,n}$:*

$$\|f - p_{2,n}\|_2 = \min_{p\in\mathbb{R}_n[x]} \|f - p\|_2.$$

Elle est caractérisée par

$$\forall p \in \mathbb{R}_n[x], \quad \langle f - p_{2,n}, p\rangle = 0.$$

Remarque 3.31. Nous avons donc $p_{2,n} = \sum_{k=0}^{n} \frac{\langle p_k, f\rangle}{\|p_k\|_2^2} p_k$.

Maintenant, énonçons le premier résultat de convergence. On l'établit pour la norme L^2.

Théorème 3.32. *Si $]a,b[$ est borné, alors pour tout $f \in \mathcal{E}(w)$, nous avons*

$$p_{2,n} \xrightarrow{\|\cdot\|_2} f \text{ quand } n \to \infty.$$

À première vue, la remarque 3.31 nous indique que le calcul de ces projections semble passer par le calcul de plusieurs intégrales. Si l'on ne dispose pas d'un autre procédé, le calcul d'une meilleure approximation L^2 est donc significativement plus coûteux que le calcul d'un polynôme d'interpolation de même degré.

Nous introduisons maintenant un outil extrêmement utile quand on fait de l'approximation polynomiale. Il nous aide à évaluer la qualité d'approximation, en norme sup, d'un polynôme ou d'une famille de polynômes.

3.3.2 Constante de Lebesgue

Sans perte de généralité, nous supposons $[a, b] = [-1, 1]$.

Définition 3.33. *On dit qu'une application linéaire $L : \mathcal{C}([-1,1]) \to \mathbb{R}_n[x]$ est une projection sur $\mathbb{R}_n[x]$ si $Lp = p$ pour tout $p \in \mathbb{R}_n[x]$. La norme d'opérateur*

$$\Lambda = \sup_{f \in \mathcal{C}([-1,1])} \frac{\|Lf\|_\infty}{\|f\|_\infty}$$

est appelée la constante de Lebesgue pour la projection.

Le résultat suivant nous permet de comparer l'erreur d'approximation d'un polynôme associée à la projection à l'erreur minimax. Il va se révéler très précieux quand on va faire jouer à l'interpolation et à la projection L^2 le rôle de la projection de la définition 3.33.

Proposition 3.34. *Soit Λ la constante de Lebesgue pour la projection L de $\mathcal{C}([-1,1])$ sur $\mathbb{R}_n[x]$. Soit $f \in \mathcal{C}([-1,1])$ et soit $p = Lf$. Soit p^* l'approximation minimax de degré au plus n de f. Alors, nous avons*

$$\|f - p\|_\infty \leqslant (1 + \Lambda) \underbrace{\|f - p^*\|_\infty}_{E_n(f)}.$$

Constante de Lebesgue pour l'interpolation polynomiale

Soient $x_0, \ldots, x_n$ des points deux-à-deux distincts de $[-1, 1]$. On considère l'opérateur d'interpolation de Lagrange

$$L_n : \mathcal{C}([-1,1]) \to \mathbb{R}_n[x], \qquad L_n f(x) = \sum_{k=0}^{n} f(x_k)\ell_k(x).$$

On démontre :

Théorème 3.35. *La constante de Lebesgue de l'interpolation de Lagrange de degré-n aux points $x_0, \ldots, x_n$ est égale à*

$$\max_{x \in [-1,1]} \sum_{k=0}^{n} |\ell_k(x)|.$$

Partant de ce résultat, on établit plusieurs inégalités clés :

Théorème 3.36. *La constante de Lebesgue $\Lambda_n = \Lambda(L_n)$ satisfait*

$$\frac{2}{\pi}\left(\log(n+1) + \gamma + \log\frac{4}{\pi}\right) \leqslant \Lambda_n, \text{ où } \frac{2}{\pi}\left(\gamma + \log\frac{4}{\pi}\right) = 0.52125\ldots$$

De plus,

— *pour les nœuds de Tchebychev (de première et seconde espèces), nous avons la borne*

$$\Lambda_n \leqslant \frac{2}{\pi}\log(n+1) + 1 \quad \textit{et} \quad \Lambda_n \sim \frac{2}{\pi}\log n \textit{ quand } n \to +\infty;$$

— *pour les points équidistants,*

$$\Lambda_n > \frac{2^{n-2}}{n^2} \quad \textit{et} \quad \Lambda_n \sim \frac{2^{n+1}}{en\log n} \textit{ quand } n \to +\infty.$$

Remarque 3.37. On déduit de ce théorème que les interpolants de Tchebychev (c.-à-d. les polynômes d'interpolation aux nœuds de Tchebychev) sont des approximations quasi-optimales :

- $\Lambda_{15} = 2.76\ldots$: on perd au plus 2 bits si l'on utilise un interpolant de Tchebychev à la place du polynôme minimax ;

- $\Lambda_{30} = 3.18\ldots$: on perd au plus 2 bits si l'on utilise un interpolant de Tchebychev à la place du polynôme minimax ;
- $\Lambda_{100} = 3.93\ldots$: on perd au plus 2 bits si l'on utilise un interpolant de Tchebychev à la place du polynôme minimax ;
- $\Lambda_{100000} = 8.32\ldots$: on perd au plus 4 bits si l'on utilise un interpolant de Tchebychev à la place du polynôme minimax ;

Constante de Lebesgue pour la meilleure approximation L_2

Définition 3.38. *Quand l'espace considéré est $\mathcal{E}\left(\frac{1}{\sqrt{1-x^2}}\right)$, la meilleure approximation polynomiale, au sens L_2, notée $p_{2,n}$ est appelée série tronquée de Tchebychev de f d'ordre n. Ses coefficients a_k sont appelés coefficients de Tchebychev de f. Ils sont donnés par*

$$a_k = \begin{cases} \frac{\langle f, T_k\rangle}{\langle T_k, T_k\rangle}, & k \neq 0, \\ \frac{1}{2}\frac{\langle f, T_0\rangle}{\langle T_0, T_0\rangle}, & k = 0, \end{cases}$$

et la série formelle

$$\sum_{k=0}^{\infty} a_k T_k(x)$$

est appelée développement de Tchebychev de f.

Théorème 3.39. *La constante de Lebesgue de l'application $f \in \mathcal{E}\left(\frac{1}{\sqrt{1-x^2}}\right) \mapsto p_{2,n}$ est*

$$\Lambda_n = \frac{1}{2\pi}\int_{-\pi}^{\pi}\left|\frac{\sin((n+1/2)t)}{\sin(t/2)}\right| \mathrm{d}t.$$

Son comportement obéit à

$$\Lambda_n \leqslant \frac{4}{\pi^2}\log(n+1) + 3 \quad \textit{et} \quad \Lambda_n \sim \frac{4}{\pi^2}\log n \textit{ quand } n \to +\infty.$$

Remarque 3.40. On déduit là aussi de ce théorème que les séries de Tchebychev tronquées sont des approximations quasi-optimales :

- $\Lambda_{15} = 4.12\ldots$: on perd au plus 3 bits si l'on utilise une série tronquée de Tchebychev à la place du polynôme minimax ;
- $\Lambda_{30} = 4.39\ldots$: on perd au plus 3 bits si l'on utilise une série tronquée de Tchebychev à la place du polynôme minimax ;
- $\Lambda_{100} = 4.87\ldots$: on perd au plus 3 bits si l'on utilise une série tronquée de Tchebychev à la place du polynôme minimax ;
- $\Lambda_{100000} = 7.66\ldots$: on perd au plus 3 bits si l'on utilise une série tronquée de Tchebychev à la place du polynôme minimax.

La combinaison du corollaire 3.6, de la proposition 3.34 et des théorèmes 3.36 et 3.39 nous donne un premier énoncé de convergence uniforme des interpolants de Tchebychev et des séries tronquées de Tchebychev vers la fonction considérée.

Corollaire 3.41. *Si f est Lipschitz continue sur $[-1, 1]$, alors*

1. *la suite des polynômes d'interpolation aux nœuds de Tchebychev converge uniformément vers f ;*
2. *la suite des séries tronquées de Tchebychev de f converge uniformément vers f.*

Nous concluons avec le théorème annoncé dans la remarque 3.5.

Théorème 3.42. *Soit f continue sur $[-1, 1]$. On note $(a_k)_{k\in\mathbb{N}}$ la suite des coefficients de la série de Tchebychev de f, $(f_n)_{n\in\mathbb{N}}$ la suite de ses séries tronquées de Tchebychev et $(p_n)_{n\in\mathbb{N}}$ la suite des interpolants de Tchebychev de f.*

1. *Les coefficients a_k tendent vers 0 quand $k \to \infty$.*

2. *Si f est Lipschitz continue sur $[-1,1]$, alors (f_n) converge absolument et uniformément vers f et (p_n) converge uniformément vers f.*
3. *Si f est $\mathcal{C}^m$ et $f^{(m)}$ est Lipschitz continue, alors $a_k = O(1/k^{m+1})$, $\|f - f_n\|_\infty = O(n^{-m})$ et $\|f - p_n\|_\infty = O(n^{-m})$.*
4. *Si f est analytique à l'intérieur de l'ellipse $\{z \in \mathbb{C}; \left|z + \sqrt{z^2-1}\right| \leqslant r\}$ avec $r > 1$, alors $a_k = O(r^{-k})$, $\|f - f_n\| = O(r^{-n})$ et $\|f - p_n\|_\infty = O(r^{-n})$.*
5. *Si f est entière, alors $a_k = O(1/k!)$, $\|f - f_n\| = O(1/n!)$ et $\|f - p_n\|_\infty = O(1/n!)$.*
6. *On note P_n^* le polynôme minimax de degré au plus n de f. Si $f \in \mathcal{C}^{n+1}([-1,1])$, il existe $\xi_1, \xi_2, \xi_3 \in]-1,1[$ tels que*

$$\|f - P_n^*\|_\infty = \frac{|f^{(n+1)}(\xi_1)|}{2^n(n+1)!}, \text{ voir [14]};$$
$$\|f - f_n\|_\infty = \frac{|f^{(n+1)}(\xi_2)|}{2^n(n+1)!}, \text{ voir [58]};$$
$$\|f - p_n\|_\infty = \frac{|f^{(n+1)}(\xi_3)|}{2^n(n+1)!}, \text{ cf. théorème 3.18 et proposition 3.19.}$$

Remarque 3.43. Le développement en série de Tchebychev de f est le développement en série de Fourier de $f \circ \cos$. Par conséquent, de nombreux résultats sur la convergence des développements en série de Tchebychev peuvent être déduits des résultats correspondants dans la théorie des séries de Fourier [223].

Première partie

Outils pour l'évaluation efficace et certifiée de fonctions

Préambule

Le problème autour duquel se concentre la majeure partie de mon activité de recherche est l'évaluation des fonctions en machine. Le cadre arithmétique dans lequel j'évolue est principalement celui de la précision finie, dans lequel les nombres ont une taille et un format contraints, tels que les nombres flottants définis par le standard IEEE 754-2008 [95, 144] qui régit l'arithmétique virgule flottante. On vise un calcul des fonctions fiable et efficace. Cela - en particulier, le calcul des fonctions élémentaires, telles que l'exponentielle, le logarithme, le sinus ou le cosinus, et les fonctions spéciales - constitue un problème central dans le domaine du calcul scientifique. On rencontre aussi ce problème en traitement du signal et des images et en contrôle numérique. Par exemple, dans le cas d'applications multimédia (lecteurs DVD), il y a besoin d'évaluer de nombreuses fonctions élémentaires.

L'évaluation d'une fonction f passe en général par celle d'une très bonne approximation p de f : cela peut être une fraction rationnelle ou une série de Fourier tronquée mais c'est le plus souvent un polynôme. Les opérations $+$, $-$ et $\times$ sont optimisées en machine et de plus, nous avons à notre disposition un nombre considérable d'outils théoriques et algorithmiques et d'implémentations fines pour manipuler les polynômes. Usuellement, on mesure les erreurs d'approximation (plaçons-nous sur un intervalle compact $[a, b]$ par exemple) à l'aide de :

— la norme sup $\|\cdot\|_{\infty,[a,b]}$ (on parlera d'approximation minimax). L'erreur sera définie par $\|f-p\|_\infty = \sup_{a\leqslant x\leqslant b} |f(x) - p(x)|$,

— la norme L^2 (on parlera d'approximation aux moindres carrés). L'erreur sera définie par

$$\|f - p\|_{2,[a,b]} = \left(\int_a^b w(x)\,(f(x) - p(x))^2\,dx\right)^{1/2},$$

où w est une *fonction poids* , ce qui pour nous signifiera qu'elle est continue sur $]a, b[$ et à valeurs strictement positives presque partout.

— la « distance relative », à savoir

$$||p - f||_{\text{rel},[a,b]} = \sup_{a\leqslant x\leqslant b} \frac{1}{|f(x)|}|p(x) - f(x)|.$$

Maintenant que nous avons défini les mesures d'erreur usuelles, nous pouvons présenter les principales étapes de la synthèse d'une routine d'évaluation, à erreur certifiée η, d'une fonction φ sur $\mathbb{R}$ ou un sous-ensemble de $\mathbb{R}$:

Étape 0. Calcul de pires cas pour l'arrondi correct.
Il s'agit de garantir à l'utilisateur que les fonctions que l'on évalue vont l'être avec arrondi correct (le nombre machine renvoyé est l'arrondi exact de la valeur de la fonction, comme si elle était calculée en précision infinie). Il faut donc arriver à déterminer sur quelle précision intermédiaire effectuer les calculs afin de garantir cet arrondi correct. Noter que cette qualité de calcul-là n'est actuellement pas visée par toutes les bibliothèques numériques, d'où ce numéro 0 pour cette étape. Je présente mon travail sur l'arrondi correct de fonctions dans le chapitre 5.

Étape 1. Réduction d'argument.
Les algorithmes utilisés dans les étapes ultérieures requièrent que l'argument appartienne à un intervalle borné et de taille raisonnable tel que $[1/2, 1]$, $[1, 2]$. De fait, l'évaluation d'une fonction élémentaire φ en un nombre flottant x se fait souvent par l'intermédiaire d'un calcul $f(x^*)$ où

— f est une fonction élémentaire, pas nécessairement différente de φ, dont l'ensemble de définition est contenu dans un intervalle $[a, b]$,
— x^* est un nombre flottant déduit de x et que l'on appelle *argument réduit* ; x^* appartient au domaine de convergence de l'algorithme implanté pour l'évaluation de f.

La réduction d'argument est l'opération qui détermine $f(x^*)$ à partir de $\varphi(x)$. Cette opération doit se faire rapidement mais avec un contrôle parfait de la précision tout au cours de ce processus.

On peut consulter [155, Chap. 11], [170, 160, 44] et **J-4** pour plus de détails.

Étape 2. Calcul d'une approximation polynomiale $p^\star$ efficace en machine.

C'est un problème crucial de pouvoir évaluer l'approximation polynomiale de la fonction f de la façon la moins coûteuse possible tout en conservant la meilleure précision possible. En effet, une fois calculé, l'approximant va être stocké en « dur » dans un logiciel ou sur du matériel et peut être appelé des milliers, voire des millions de fois par la suite. Il est donc très important que son implantation, que son évaluation requièrent le moins de temps, de mémoire, de silicium possibles et qu'en même temps, il fournisse, avec garantie, une approximation de la fonction à l'erreur souhaitée près. Le type d'approximants avec lesquels on peut travailler est particulier : les coefficients doivent respecter des contraintes imposées par le support ou l'application visés, telles qu'une limitation de leur taille en bits par exemple. Les méthodes classiques de détermination de bons approximants ne peuvent pas s'appliquer ici (les approximants qu'elles fournissent sont en général à coefficients réels quelconques) et il faut donc concevoir de nouveaux procédés pour obtenir de bons approximants qui soient de la forme désirée. Cela fera l'objet du chapitre 4, où je parlerai aussi du même problème dans les cadres de l'approximation rationnelle et du traitement du signal.

Étape 3. Calcul d'une erreur d'approximation rigoureuse $\|f - p^\star\|$ ou $\|1 - p^\star/f\|$.

Pour nombre des applications que nous visons, nous souhaitons garantir à l'utilisateur que les fonctions que l'on évalue vont l'être avec arrondi correct ou, au moins, avec arrondi fidèle (le nombre machine renvoyé est soit l'arrondi correct de la valeur de la fonction, soit un nombre machine, prédécesseur ou successeur de l'arrondi correct, tel que la valeur de la fonction, calculée en précision infinie, soit comprise entre ce nombre machine et l'arrondi correct). Cela impose la certification des calculs, soit au sens de l'arithmétique d'intervalles soit au sens de la preuve formelle. Cela m'a amené à m'intéresser depuis 2008 au calcul fin et certifié de l'erreur d'approximation c.-à-d. la norme sup de la différence entre la fonction et son approximant. Depuis, ce sujet a débordé son objectif initial de validation en vue de l'arrondi correct et je me suis lancé dans un travail à long terme sur le calcul certifié d'approximations polynomiales de solutions de certaines équations différentielles. Ces questions feront l'objet du chapitre 6.

Étape 4. Calcul d'une erreur d'évaluation certifiée pour $p^\star$.

L'outil Gappa [4] s'en charge, en fournissant même une preuve formelle en COQ. On peut consulter [147, 45, 47] pour plus de détails.

Dans cette première partie, nous allons donc étudier un peu plus en détail les étapes 0, 2, 3 et des problèmes qui y sont liés.

4. http://gappa.gforge.inria.fr/

Chapitre 4

Approximation efficace en machine

Quand on cherche à déterminer de très bons (voire les meilleurs) approximants à coefficients réels et s'exprimant dans des bases standard (pour les polynômes, penser à $\{1, x, x^2, \ldots, x^n\}$ ou aux bases de polynômes orthogonaux), on dispose de méthodes classiques qui sont satisfaisantes. Par exemple, on calcule les approximations polynomiales aux moindres carrés à l'aide d'une méthode de projection utilisant des polynômes orthogonaux. Les approximations polynomiales minimax peuvent être obtenues par l'algorithme de Remez, qui se généralise au cas des fractions rationnelles et des sommes de cosinus. D'autres procédés (calcul de polynômes de Taylor, d'interpolation aux nœuds de Tchebychev, ...) permettent d'obtenir de bons approximants polynomiaux pour $||\cdot||_{\infty,[a,b]}$. On trouvera dans [31, 175] une présentation de ces algorithmes.

Malheureusement, ces méthodes théoriques ou pratiques fournissent seulement des solutions en précision arbitraire, comme dans l'algorithme de Remez par exemple, qui ne sont pas adaptées aux machines. En effet, les ordinateurs ne peuvent stocker de manière exacte et évaluer efficacement et précisément que des polynômes dont les coefficients sont des nombres rationnels dont le dénominateur est une puissance d'un entier b fixé ($b = 2$ le plus souvent) et qui a de plus, tout comme le numérateur, une taille limitée.

Le problème soulevé est donc de trouver de très bons approximants respectant des contraintes sur la taille en nombre de bits des coefficients. J'ai rencontré cette question à travers les problématiques d'évaluation de fonction mais elle est en fait une question déjà ancienne en traitement du signal (on parle, par exemple, de quantification des coefficients d'un filtre [168, 10, 215]). L'idée la plus simple pour l'attaquer est de considérer l'approximation minimax (resp. aux moindres carrés) fournie par les méthodes classiques et d'arrondir chaque coefficient suivant la contrainte en nombre de bits imposée par l'application. Cependant, l'approximation ainsi obtenue n'est en général pas le meilleur (ni même parfois un bon) approximant minimax (resp. aux moindres carrés) parmi les approximants satisfaisant à la contrainte de taille en nombre de bits sur les coefficients. Il s'agit donc de produire ce meilleur (ou un très bon, suivant l'application visée) approximant sous cette contrainte. Le gain peut être significatif, comme l'exemple présenté dans la sous-section 4.5.3 l'illustre.

Je me propose de présenter dans ce chapitre une partie de mes travaux en cours ou achevés sur cette problématique. Je commencerai par formuler dans la section 4.1 le problème général qui nous intéresse dans le cas de l'approximation en norme sup, puis je présenterai (une partie de) l'approche que j'en ai proposée avec mes collègues, via l'algorithmique des réseaux euclidiens dans la section 4.2. Cette présentation doit se décliner suivant les applications visées. Je compléterai donc la description de l'approche en traitant plus en détail le cas où l'approximant est :

— un polynôme, ce qui constitue la situation la plus fréquente dans le cadre de l'évaluation de fonctions élémentaires. Les résultats, associés à **J-6, J-8, C-1, C-4, C-5, S-6**, sont présentés dans la section 4.5 ;

— une réponse fréquentielle dans le cas d'un filtre à réponse impulsionnelle finie (RIF), voir la section 4.6 qui évoque **S-3**.

Je conclurai en évoquant brièvement des résultats obtenus dans le cas de l'approximation polynomiale

en norme L^2 dans la section 4.5.4, et dans le cas de l'approximation par une une fraction rationnelle (ici, le but est de produire des opérateurs matériels d'évaluation extrêmement efficaces, voir 4.7).

Ces travaux ont été réalisés avec mes collègues Sylvain Chevillard, Miloš Ercegovac, Silviu-Ioan Filip, Guillaume Hanrot, Jean-Michel Muller, Arnaud Tisserand et Serge Torres.

Le contenu de la section 4.5 est au cœur de la thèse de Sylvain Chevillard [32] et celui de la section 4.6 au cœur de la thèse de Silviu-Ioan Filip [66].

4.1 Formulation générale du problème

Soit X un ensemble compact de $\mathbb{R}$ et $n \in \mathbb{N}$, on considère une fonction $f \in \mathcal{C}(X)$, le $\mathbb{R}$-espace vectoriel des fonctions continues sur X, et une famille $\{\varphi_0, \ldots, \varphi_n\}$ d'éléments de $\mathcal{C}(X)$ linéairement indépendants sur $\mathbb{R}$. On appelle polynôme généralisé un élément de $\mathrm{Vect}_{\mathbb{R}}\{\varphi_0, \ldots, \varphi_n\}$. La question qui nous préoccupe est celle de l'approximation de f à l'aide de $\{\varphi_0, \ldots, \varphi_n\}$ et de coefficients respectant certaines contraintes. En amont, le problème classique est celui de l'approximation minimax :

Problème 4.1 (Approximation minimax)**.** *Étant donnée une fonction poids W [1], déterminer un polynôme généralisé $P^* \in \mathrm{Vect}_{\mathbb{R}}\{\varphi_0, \ldots, \varphi_n\}$ tel que l'erreur pondérée $E^* = W\,(P^* - f)$, ait une norme sup minimale :*

$$\|E^*\|_{\infty,X} = \min_{P \in \mathrm{Vect}_{\mathbb{R}}\{\varphi_0,\ldots,\varphi_n\}} \max_{x \in X} |W(x)\,(P(x) - f(x))|\,.$$

Ce problème admet toujours une solution. Si nous supposons de plus que $\{\varphi_0, \ldots, \varphi_n\}$ vérifie la condition de Haar (présentée dans le chapitre 3), alors on sait que cette solution est unique et caractérisée par le théorème d'équioscillation [31, Ch. 3.4] :

Théorème 4.2 (Théorème d'équioscillation - Kirchberger, 1902)**.** *Une condition nécessaire et suffisante pour que P soit l'unique élément de $\mathrm{Vect}_{\mathbb{R}}\{\varphi_0, \ldots, \varphi_n\}$ qui minimise l'erreur pondérée $\delta_X = \|E\|_{\infty,X}$ est qu'il existe au moins $n+2$ valeurs x_k dans X telles que $x_0 < x_1 < \ldots < x_{n+1}$ et*

$$E(x_k) = -E(x_{k+1}) = \lambda(-1)^k \delta_X, \qquad k = 0, \ldots, n,$$

où $\lambda \in \{\pm 1\}$ est fixée.

La programmation linéaire réelle [188] permet de calculer la solution (ou plus précisément, une approximation aussi bonne que souhaité de la solution) au problème 4.1. Toutefois, lorsque la condition de Haar est vérifiée, la méthode de prédilection pour ce calcul est l'algorithme de Remez et ses variantes telles que l'algorithme de Parks-McClellan pour la synthèse de filtres RIF. Noter qu'un algorithme dû à Descloux [50, 51] permet de continuer à utiliser une méthode d'échange, comme celle de l'algorithme de Remez même lorsque la condition de Haar n'est pas vérifiée : il n'y a alors aucune garantie de convergence mais nous avons observé son bon comportement en pratique lors du travail de thèse de S. Chevillard.

Si l'on impose des contraintes de précision finie, par exemple, sur les coefficients des φ_k, la question devient en général bien plus difficile à résoudre. Le type de problèmes qui nous intéresse dans ce chapitre peut être modélisé de la façon suivante :

Problème 4.3 (Approximation minimax en précision finie)**.** *Soit X un ensemble compact de $\mathbb{R}$, $n \in \mathbb{N}$, soient $f \in \mathcal{C}(X)$ et $\{\varphi_0, \ldots, \varphi_n\}$ une famille d'éléments de $\mathcal{C}(X)$ linéairement indépendants sur $\mathbb{R}$, W une fonction poids, déterminer des $a_i \in \mathbb{Z}$ tels que*

$$P = a_0\varphi_0 + a_1\varphi_1 + \cdots + a_n\varphi_n$$

minimise $\|W\,(P - f)\,\|_{\infty,X}$.

Dans la suite, pour désigner les approximants intervenant dans ce problème, on parlera d'approximants en précision finie ou quantifiés. Le problème 4.3 a toujours une solution, mais cette fois-ci, elle n'a aucune raison d'être unique, même lorsque la condition de Haar est vérifiée. Pour aider notre lectrice à s'approprier cette question, nous allons traiter deux exemples-jouets. Ils n'ont d'autre intérêt que pédagogique.

1. Ici, W est supposée continue et à valeurs strictement positives sur X.

Exemple 4.4. Nous nous plaçons dans le cadre de l'étude menée dans la section 4.5. On souhaite approcher la fonction $f : x \mapsto \sqrt{2} + \pi x + ex^2$ sur l'intervalle $[2,\ 4]$ par un polynôme de degré au plus 2 avec des coefficients au format binary64. L'approximation minimax est la fonction f elle-même. Si l'on arrondit chaque coefficient de f vers le flottant binary64 le plus proche (nous parlerons alors d'« arrondi naïf »), on obtient le polynôme

$$\widehat{P} = \frac{6369051672525773}{4503599627370496} + \frac{884279719003555}{281474976710656}x + \frac{6121026514868073}{2251799813685248}x^2$$

et nous avons $\|f - \widehat{P}\|_\infty \simeq 2.70622 \cdot 10^{-15}$. Mais le meilleur approximant polynomial de degré au plus 2 et à coefficients au format binary64 est

$$P^* = \frac{6369051672525769}{4503599627370496} + \frac{3537118876014221}{1125899906842624}x + \frac{6121026514868073}{2251799813685248}x^2$$

pour lequel $\|f - P^*\|_\infty \simeq 2.2243 \cdot 10^{-16}$: on constate qu'il y a un facteur 10 entre l'erreur d'approximation optimale et l'erreur naïve.

Exemple 4.5. À l'instar de ce que nous développerons dans la section 4.6, nous souhaitons calculer un filtre RIF passe-bas de longueur 5, avec des coefficients virgule fixe sur 5 bits et une bande passante (passband en anglais) $[0, 0.4\pi]$ et une bande affaiblie (stopband en anglais) $[0.5\pi, \pi]$ uniformément pondérées : nous posons $D(\omega) = 1$ pour $\omega \in [0, 0.4\pi]$ et $D(\omega) = 0$ si $\omega \in [0.5\pi, \pi]$ et nous cherchons à déterminer a_0, a_1 et $a_2 \in \{-2^4, \ldots, 2^4 - 1\}$ tels que l'erreur d'approximation

$$\max_{\omega \in [0, 0.4\pi] \cup [0.5\pi, \pi]} \left| \frac{a_0}{2^4} + \frac{a_1}{2^3}\cos(\omega) + \frac{a_2}{2^3}\cos(2\omega) - D(\omega) \right|$$

soit aussi petite que possible.

L'algorithme de Parks-McClellan [169] retourne une réponse fréquentielle en précision infinie $H^*(\omega) = 0.430\ldots + 0.860\ldots\cos(\omega) + 0.069\ldots\cos(2\omega)$ et une erreur d'approximation $E^* = 0.360\ldots$. Si nous arrondissons chacun de ses coefficients vers le nombre de la forme $k/2^3$ (ou $k/2^4$ pour le coefficient constant) le plus proche, nous obtenons $H_{\text{naïve}}(\omega) = \frac{7}{2^4} + \frac{7}{2^3}\cos(\omega) + \frac{1}{2^3}\cos(2\omega)$, et une erreur $E_{\text{naïve}} = 0.4375$. L'approche que je présente ici retourne $H(\omega) = \frac{8}{2^4} + \frac{6}{2^3}\cos(\omega) + \frac{1}{2^3}\cos(2\omega)$, qui donne une erreur $E = 0.375$ meilleure. On peut montrer que H est en fait une réponse fréquentielle optimale H_{opt} et E est une erreur d'approximation optimale E_{opt}.

Cet exemple permet d'illustrer deux faits intéressants :

— l'approche immédiate (l'« arrondi naïf »), consistant à arrondir les coefficients de la réponse fréquentielle en précision infinie vers les coefficients les plus proches parmi ceux satisfaisant les contraintes imposées, peut, même dans des cas très simples, donner des résultats qui sont loin de l'optimal (par exemple, [112] mentionne un exemple pour lequel on observe une amélioration de 30 dB).

— Une autre idée naturelle est de considérer toutes les réponses fréquentielles que l'on obtient en arrondissant vers $+\infty$ ou $-\infty$ les coefficients de la réponse fréquentielle en précision infinie. Un (gros) défaut immédiat de cette approche est que le nombre de possibilités est exponentiel en le degré du filtre. De plus, notre exemple-jouet montre qu'il est possible qu'aucune de ces réponses fréquentielles ne fournisse une réponse optimale ! Ce fait avait déjà été noté dans [132].

Comment attaque t-on donc ce problème 4.3 d'une manière non triviale ? On peut le reformuler naturellement comme une question d'optimisation :

$$\begin{aligned}
&\text{minimiser} && \delta \\
&\text{sous les conditions} && W(x)\left(\sum_{k=0}^{n} a_k \varphi_k(\omega) - f(x)\right) \leqslant \delta,\ x \in X, \\
& && W(x)\left(f(x) - \sum_{k=0}^{n} a_k \varphi_k(\omega)\right) \leqslant \delta,\ x \in X, \\
& && a_k \in \mathbb{Z}, k = 0, \ldots, n.
\end{aligned} \tag{4.1}$$

Une approche immédiate consiste à remplacer X par une discrétisation, ce qui donne lieu à un problème de programmation linéaire entière mixte (PLEM), que l'on essaie de résoudre avec les outils actuels Gurobi [2], ISL [3], PIP [4], SCIP [5]. Cette approche est malheureusement vite limitée, même si l'on essaie de travailler avec une discrétisation parcimonieuse : les degrés ou les formats des coefficients dans nombre des applications que nous traitons rendent ces instances de PLEM bien trop difficiles. J'ai donc proposé, d'abord avec S. Chevillard puis avec Silviu Filip et Guillaume Hanrot, une approche via l'algorithmique des réseaux euclidiens. Comme nous allons le voir, ce procédé est à la fois efficace (en temps et en mémoire) et robuste (elle permet de traiter des problèmes de taille totalement inaccessible à ce jour par les méthodes de PLEM).

Mentionnons que l'on trouve dans [216, 217, 218] l'utilisation de l'algorithme LLL dans un esprit proche du nôtre. Le problème du calcul de polynômes de Tchebychev entiers y est amorcé par la résolution d'un SVP à l'aide de LLL.

4.2 Approche via la réduction de réseaux euclidiens

À l'instar de l'approche par la PLEM, nous commençons par discrétiser le problème 4.3 : nous choisissons $\ell(\geqslant n+1)$ points $x_0, \ldots, x_{\ell-1}$ de X et nous allons donc tenter de déterminer un polynôme généralisé $P = \sum_{k=0}^{n} a_k \varphi_k$, avec $a_k \in \mathbb{Z}$ pour $k = 0, \ldots, n$, tel que l'erreur

$$\sup_{j=0,\ldots,\ell-1} |W(x_j)\,(P(x_j) - f(x_j))| \tag{4.2}$$

soit aussi petite que possible. Autrement dit, nous voulons trouver $(a_0, \ldots, a_n) \in \mathbb{Z}^{n+1}$ tel que les vecteurs

$$\begin{pmatrix} W(x_0)\sum_{k=0}^{n} a_k\varphi_k(x_0) \\ W(x_1)\sum_{k=0}^{n} a_k\varphi_k(x_1) \\ \vdots \\ W(x_{\ell-1})\sum_{k=0}^{n} a_k\varphi_k(x_{\ell-1}) \end{pmatrix} \text{ et } \begin{pmatrix} W(x_0)f(x_0) \\ W(x_1)f(x_1) \\ \vdots \\ W(x_{\ell-1})f(x_{\ell-1}) \end{pmatrix}$$

c.-à-d.

$$a_0 \underbrace{\begin{pmatrix} W(x_0)\varphi_0(x_0) \\ W(x_1)\varphi_0(x_1) \\ \vdots \\ W(x_{\ell-1})\varphi_0(x_{\ell-1}) \end{pmatrix}}_{\mathbf{b}_0} + \cdots + a_n \underbrace{\begin{pmatrix} W(x_0)\varphi_n(x_0) \\ W(x_1)\varphi_n(x_1) \\ \vdots \\ W(x_{\ell-1})\varphi_n(x_{\ell-1}) \end{pmatrix}}_{\mathbf{b}_n} \text{ et } \underbrace{\begin{pmatrix} W(x_0)f(x_0) \\ W(x_1)f(x_1) \\ \vdots \\ W(x_{\ell-1})f(x_{\ell-1}) \end{pmatrix}}_{\mathbf{t}}$$

soient aussi proches que possible pour la norme $\|\cdot\|_\infty$. Cela revient donc à

$$\text{minimiser } \|a_0\mathbf{b}_0 + \ldots + a_n\mathbf{b}_n - \mathbf{t}\|_\infty, a_0, \ldots, a_n \in \mathbb{Z} \tag{4.3}$$

qui est une instance de CVP$_\infty$, le réseau euclidien utilisé étant $\mathbb{Z}\mathbf{b}_0 + \ldots + \mathbb{Z}\mathbf{b}_n \subset \mathbb{R}^\ell$.

À ce stade, on constate que nous n'avons procédé qu'à des reformulations du problème initial et il reste encore plusieurs questions importantes en suspens :

— quelle discrétisation choisir (nombre de points ℓ, localisation de ces points) ? Cette étape cruciale va dépendre des applications, comme nous allons le voir dans les sections 4.5 et 4.6, dans lesquelles nous rapporterons les solutions qui nous ont paru les mieux adaptées.
— Comment résoudre le CVP$_\infty$ rencontré en (4.3) ? Cela fera l'objet de la section suivante.
— Compliquons un peu notre problème : que faire si l'on souhaite déterminer des coefficients flottants plutôt que des coefficients entiers ? C'est le cas lorsque l'on cherche un polynôme à coefficients flottants pour l'évaluation de fonction par exemple. Nous présenterons la solution que nous proposons dans la section 4.5.2.

2. http://www.gurobi.com/
3. http://isl.gforge.inria.fr/
4. http://www.piplib.org/
5. http://scip.zib.de/

Remarque 4.6. En pratique, il est souvent pertinent d'essayer aussi d'approcher P^*, l'approximation minimax, à la place de la fonction f : dans ce cas, nous considérons $\mathbf{t} = (W(x_0)P^*(x_0) \cdots W(x_{\ell-1})P^*(x_{\ell-1}))^t$ dans (4.3). L'idée ici est de demander à notre polynôme d'imiter du mieux possible la meilleure approximation polynomiale.

4.3 Résolution (approchée) du CVP$_\infty$

Nous résolvons de manière approchée (4.3) en deux temps :

1. même si nous avons vu dans le chapitre 2 que les problèmes CVP$_2$ et CVP$_\infty$ sont tous deux NP-difficiles, l'existence d'algorithmes dans le cadre euclidien qui sont à la fois pratiques et implémentés dans des bibliothèques très performantes nous ont amené à essayer de résoudre (de manière approchée) plutôt la version $\|\cdot\|_2$ de (4.3) :

$$\text{minimiser } \|a_0\mathbf{b}_0 + \ldots + a_n\mathbf{b}_n - \mathbf{t}\|_2, a_0, \ldots, a_n \in \mathbb{Z}. \tag{4.4}$$

   Dans la section 4.5, nous utilisons l'algorithme 4 (celui du plan le plus proche de Babai) qui se comporte de manière très satisfaisante. Pour la section 4.6, nous avons sélectionné l'algorithme 5 (le plongement de Kannan) en nous autorisant à brancher en entrée les algorithmes LLL, BKZ et HKZ, cf. chapitre 2. Nous obtenons donc :
   — un vecteur $\sum_{k=0}^{n} a_k\mathbf{b}_k$ proche de $\mathbf{t}$ pour $\|\cdot\|_2$ et, espérons-le, proche de $\mathbf{t}$ pour $\|\cdot\|_\infty$,
   — une base réduite $(\mathbf{c}_0, \ldots, \mathbf{c}_n)$ de $(\mathbf{b}_0, \ldots, \mathbf{b}_n)$ (en appliquant soit LLL, soit LLL, BKZ ou HKZ) et la matrice de changement de base $\mathbf{U}$.

2. en utilisant des combinaisons des vecteurs courts $(\mathbf{c}_0, \ldots, \mathbf{c}_n)$ calculés au cours de la première étape, nous « tournons » autour de la solution approchée de l'instance de CVP$_2$ (4.4) afin d'améliorer la solution approchée de l'instance de CVP$_\infty$ (4.3). L'algorithme 8 donne un exemple de cette recherche au voisinage. On y combine deux vecteurs de la base réduite, dans des directions d_0 et d_1.
   En pratique, nous limitons la valeur de la profondeur d'exploration (et donc de d_0) à une petite constante (huit dans le cadre de la section 4.6 par exemple) :
   — afin de garder un temps d'exécution raisonnable : la ligne 11 prend $O(n^2)$ opérations, et elle est exécutée $O(n)$ fois, cf. lignes 3, 4, 6 ;
   — mais aussi parce qu'en pratique nous n'obtenons, en général, pas d'amélioration significative si l'on élargit la recherche.
   Monter à trois le nombre de directions de recherche peut parfois aider mais l'augmentation du coût de calcul devient pénalisante.

Le fait de passer par la résolution de (4.4) au lieu de procéder à une attaque directe de (4.3) a une conséquence importante pour la discrétisation que l'on considère. Comme nous avons $\|\cdot\|_\infty \leqslant \|\cdot\|_2 \leqslant \sqrt{\ell}\|\cdot\|_\infty$ dans $\mathbb{R}^\ell$, il semble judicieux de prendre ℓ aussi proche que possible de sa valeur minimale $n+1$, cela afin que les vecteurs courts pour $\|\cdot\|_2$ le soient aussi pour $\|\cdot\|_\infty$.

4.4 Une synthèse de notre approche sous forme de pseudo-code

On présente dans l'algorithme 9 notre approche pour traiter le problème 4.3.

4.5 Approximation polynomiale sous contraintes sur les coefficients

Il s'agit ici de fournir les meilleures approximations polynomiales d'une fonction en exigeant par exemple que les valeurs de certains coefficients soient fixées (on peut demander que les polynômes d'approximation soient pairs ou impairs) ou que les coefficients soient des nombres flottants ou la somme de deux nombres flottants ou encore que leur écriture binaire comporte le plus de zéros possibles afin de pouvoir construire des multiplieurs les moins coûteux possibles.

Algorithme 8 Recherche au voisinage

Entrée : vecteur de coefficients $\mathbf{a} \in \mathbb{Z}^{n+1}$, une fonction f, un poids W, une base de fonctions $(\varphi_k)_{0 \leqslant k \leqslant n}$, la matrice de changement de base $\mathbf{U} \in \mathbb{Z}^{(n+1)\times(n+1)}$ donnant la base réduite, un entier positif pr (il donne la profondeur d'exploration).

Sortie : un vecteur de coefficients $\mathbf{a}' \in \mathbb{Z}^{n+1}$ tel que $\sum_{k=0}^{n} a'_k \varphi_k$ est le meilleur approximant quantifié dans un voisinage de $\sum_{k=0}^{n} a_k \varphi_k$.

// Enregistrer l'erreur d'approximation et les coefficients initiaux
$E_{\min} \leftarrow \|W(x)\,(\sum_{k=0}^{n} a_k \varphi_k(x) - f(x))\|_{X,\infty}$
$\mathbf{a}' \leftarrow \mathbf{a}$
// On tente d'améliorer cette erreur initiale en sélectionnant
// deux directions d_0 et d_1 dans lesquelles chercher
pour $d_0 = 0$ **à** pr **faire**
  pour $d_1 = d_0 + 1$ **à** n **faire**
    $\mathbf{u} \leftarrow \mathbf{U}_{0:n,d_0}, \mathbf{v} \leftarrow \mathbf{U}_{0:n,d_1}$
    // Mettre à jour les coefficients dans ces directions si
    // cela mène à une erreur d'approximation plus petite
    pour $\varepsilon_0, \varepsilon_1 \in \{0, \pm 1\}$ **faire**
      $\mathbf{a}'' \leftarrow \mathbf{a}$
      pour $i = 0$ **à** n **faire**
        $a''_i \leftarrow a''_i + u_i \varepsilon_0 + v_i \varepsilon_1$
      fin pour
      $E \leftarrow \|W(x)\,(\sum_{k=0}^{n} a''_k \varphi_k(x) - f(x))\|_{X,\infty}$
      si $E < E_{\min}$ **alors**
        $E_{\min} \leftarrow E$
        $\mathbf{a}' \leftarrow \mathbf{a}''$
      fin si
    fin pour
  fin pour
fin pour

Algorithme 9 Quantification des coefficients à l'aide de la réduction de réseaux

Entrée : Une fonction f, un degré n, une base $(\varphi_0, \ldots, \varphi_n)$, une discrétisation à ℓ points $\{x_0, \ldots, x_{\ell-1}\}$ de X ($\ell \geqslant n+1$), une approximation minimax P^*, un poids W, un algorithme de réduction de base LBR $\in$ {LLL, BKZ, HKZ}.

Sortie : un $n+1$-uplet $(a_k)_{k=0,\ldots,n} \in \mathbb{Z}^{n+1}$ tel que $\sum_{k=0}^{n} a_k \varphi_k$ est, heuristiquement, une bonne approximation de f.

// Construire les vecteurs de la base du réseau
1: **pour** $i = 0$ **à** n **faire**
2: $\quad \mathbf{b}_i \leftarrow (W(x_0)\varphi_i(x_0) \cdots W(x_{\ell-1})\varphi_i(x_{\ell-1}))^t$
3: **fin pour**
// Construire deux vecteurs cibles possibles
4: $\mathbf{t} \leftarrow (W(x_0)P^*(x_0) \cdots W(x_{\ell-1})P^*(x_{\ell-1}))^t$
5: $\mathbf{t}' \leftarrow (W(x_0)f(x_0) \cdots W(x_{\ell-1})f(x_{\ell-1}))^t$
// Calculer des solutions approchées au CVP à l'aide de l'algorithme 4
// (évaluation de fonctions) ou à l'aide de l'algorithme 5 (synthèse de filtres RIF)
6: $(\mathbf{a}, \mathbf{U}) \leftarrow$ ApproxCVP($\mathbf{B}, \mathbf{t}$, LBR)
7: $(\mathbf{a}', \mathbf{U}') \leftarrow$ ApproxCVP($\mathbf{B}, \mathbf{t}'$, LBR)
// Utiliser la recherche au voisinage de l'algorithme 8
// pour améliorer la qualité de la solution
8: $\mathbf{a} \leftarrow$ RechVoisinage($\mathbf{a}, f, W, (\varphi_k)_{0 \leqslant k \leqslant n}, \mathbf{U}$)
9: $\mathbf{a}' \leftarrow$ RechVoisinage($\mathbf{a}', f, W, (\varphi_k)_{0 \leqslant k \leqslant n}, \mathbf{U}'$)
10: $E_1 \leftarrow \|W(x)\left(\sum_{k=0}^{n} a_k \varphi_k(x) - f(x)\right)\|_{x,\infty}$
11: $E_2 \leftarrow \|W(x)\left(\sum_{k=0}^{n} a_k \varphi_k(x) - f(x)\right)\|_{x,\infty}$
12: **si** $E_2 < E_1$ **alors**
13: $\quad \mathbf{a} \leftarrow \mathbf{a}'$
14: **fin si**

4.5.1 Introduction

Lorsqu'on évalue une fonction mathématique en machine, que ce soit en logiciel ou en matériel, on remplace fréquemment la fonction par une bonne approximation. Cela est dû en partie au fait qu'en arithmétique virgule flottante, l'addition, la soustraction et la multiplication sont implémentées efficacement et avec grand soin dans les processeurs modernes, et aussi parce que l'on peut utiliser tout un arsenal de schémas efficaces d'évaluation de polynômes. De fait, de nombreux algorithmes d'évaluation de fonctions en précision finie utilisent de très bonnes approximations polynomiales [143, 155, 200, 40].

Par conséquent, la première étape pour quelqu'un qui souhaite évaluer une fonction f en précision finie est d'obtenir un polynôme p :

— qui soit suffisamment proche de f pour la mesure d'erreur requise par l'application visée,
— qui puisse être stocké et utilisé directement par l'outil qui va mettre en œuvre l'application.

Nous partons d'une fonction continue à valeurs réelles f, un intervalle $[a, b]$ et un entier $n \in \mathbb{N}$. On note $\mathbb{R}_n[X]$ le $\mathbb{R}$-espace vectoriel des polynômes à coefficients réels et de degré au plus n. Le (ou un) polynôme $p \in \mathbb{R}_n[X]$ qui approche de manière optimale f dépend bien sûr du choix fait pour mesurer la qualité de l'approximation. Dans ce qui suit, nous allons nous focaliser sur la norme sup que nous noterons $||\cdot||_{[a,b]}$ au lieu de $||\cdot||_{\infty,[a,b]}$. Elle nous permettra d'examiner les cas de l'erreur absolue et de l'erreur relative (il suffira de considérer le poids $1/f$). Nous évoquerons rapidement le cas de la norme L^2 dans la sous-section 4.5.4.

Comme nous l'avons vu dans le chapitre 3, on sait obtenir efficacement des approximations en précision infinie optimales ou quasi-optimales au sens L^∞ et au sens L^2. Le problème pour la scientifique, l'ingénieur qui implémente en logiciel ou en matériel des approximations polynomiales est qu'elle doit utiliser une arithmétique en précision finie :

— de l'arithmétique virgule fixe ou de l'arithmétique virgule flottante binary32 pour des applications matérielles, des systèmes embarqués. Il s'agit là de réduite le coût (en temps, en surface de silicium) tout en conservant une précision raisonnable,

— de l'arithmétique virgule flottante binary32 ou binary64 pour des applications logicielles. Ici, on souhaite obtenir une très grande précision tout en gardant un coût acceptable en temps et en mémoire.

Malheureusement, la plupart du temps, il n'y a aucune raison pour que les coefficients des approximations (minimax, interpolants de Tchebychev, séries de Tchebychev tronquées, etc.) évoquées dans le chapitre 3 soient représentables exactement sur un nombre fini de bits. La solution évidente, que nous avons appelée précédemment « arrondi naïf », est en général peu satisfaisante.

Explicitons maintenant le lien avec le problème 4.3. On considère une famille $\{\psi_0, \ldots, \psi_n\}$, linéairement indépendante sur $\mathbb{R}$, de fonctions continues sur $[a, b]$. Bien sûr, nous avons $\{1, x, x^2, \ldots, x^n\}$ en tête, mais nous pouvons considérer toute base de $\mathbb{R}_n[x]$, telle que $\{T_{[a,b],0}(x), \ldots, T_{[a,b],n}(x)\}$, les polynômes de Tchebychev de première espèce sur $[a, b]$. Il est aussi parfois commode de travailler avec une base incomplète comme $\{x^{i_k}\}_{0 \leqslant i_0 \leqslant \cdots \leqslant i_n}$ par exemple.

Si l'on travaille en virgule fixe, notre problème va donc être de déterminer un $(n+1)$-uplet $(a_0, \ldots, a_n) \in \mathbb{Z}^{n+1}$ tel que $P = \frac{a_0}{2^{m_0}}\psi_0 + \frac{a_1}{2^{m_1}}\psi_1 + \cdots + \frac{a_n}{2^{m_n}}\psi_n$, avec $m_0, \ldots, m_n \in \mathbb{Z}$ fixés soit aussi proche que possible de f pour la norme sup sur $[a, b]$. On pose alors $\varphi_k = \psi_k/2^{m_k}$ et nous avons donc exactement une instance du problème 4.3.

Si l'on travaille en virgule flottante, il y a une petite difficulté supplémentaire qui provient du fait que les exposants m_k ne sont pas fixés initialement. Nous allons expliquer dans 4.5.2 comment nous arrivons à déterminer en amont les valeurs de ces exposants. Cela nous permet de nous retrouver dans le cadre de la virgule fixe, et par conséquent, du problème 4.3.

En 2005, J.-M. Muller, A. Tisserand et moi-même avons proposé dans **J-6** une première méthode générale pour attaquer le problème 4.3. Elle consiste à construire avec soin un certain polytope rationnel contenant tous les $(a_0, \ldots, a_n) \in \mathbb{Z}^{n+1}$ solutions et aussi peu que possible d'autres éléments de $\mathbb{Z}^{n+1}$, puis dans un second temps, à parcourir, à l'aide de la programmation linéaire rationnelle paramétrique, tous les points à coordonnées entières appartenant à ce polytope.

Les résultats présentés dans **J-6** ont été implantés dans la bibliothèque `MEPLib`[6] (Machine-Efficient Polynomials Library) par A. Tisserand, S. Torres (LIP, AriC) et moi-même. Cette bibliothèque calcule des approximants polynomiaux sous diverses contraintes, imposées par l'utilisateur, sur les coefficients du polynôme. Cette bibliothèque s'appuie sur les bibliothèques GMP[7], MPFR[8] (arithmétique multiprécision) et PIP[9] (programmation linéaire).

L'implantation actuelle, soignée, de `MEPLib` nous permet d'atteindre des degrés pouvant aller jusqu'à 8-10. C'est amplement suffisant pour la plupart des applications matérielles (nous l'avons utilisée avec succès dans **C-1** et **J-8**) mais insuffisant pour un certain nombre d'applications logicielles. Un exemple important pour notre équipe, la bibliothèque `CRlibm`[10] [123], fournit des implantations en double précision IEEE des fonctions élémentaires du standard C99 avec une qualité de calcul supérieure (les résultats sont correctement arrondis, quelque soit le mode d'arrondi considéré) tout en conservant des performances similaires à celle des bibliothèques mathématiques concurrentes. Elle requiert des approximations polynomiales optimisées de degré 10 à 25. De même, afin de pouvoir travailler sur de grands intervalles, une équipe de chercheurs d'Intel a indiqué il y a quelques années utiliser des approximants polynomiaux impairs de degré 47 pour l'implantation de la fonction $\arctan$ sur l'Itanium.

Nous avons donc cherché à développer d'autres méthodes nous permettant de travailler sur des degrés plus grands. Pour cela, nous avons utilisé la notion de réseau euclidien et les problèmes algorithmiques qui y sont attachés (recherche du vecteur le plus court, recherche du vecteur le plus proche) pour modéliser le problème abordé dans cette action et pouvoir l'attaquer de manière efficace en pratique. Ainsi, dans **C-4**, S. Chevillard et moi-même avons utilisé l'algorithme de réduction de réseaux LLL [129] pour proposer une méthode efficace de calcul de très bons approximants polynomiaux dans le cas de la norme sup : cette méthode est heuristique au sens où si la théorie ne garantit qu'une qualité médiocre des approximants polynomiaux obtenus, ces derniers sont en pratique très bons, voire parfois optimaux. Il est même possible de la rendre rigoureuse, à l'aide de la programmation linéaire, cf. section 4.5.2. Ce

6. http://lipforge.ens-lyon.fr/projects/meplib
7. https://gmplib.org/
8. http://www.mpfr.org/
9. http://www.piplib.org/
10. https://gforge.inria.fr/scm/browser.php?group_id=5929&extra=crlibm

travail est à l'origine de tout ce que nous présentons dans les sections 4.1 à 4.5.3.

Les résultats présentés dans **C-4** sont implantés dans la bibliothèque `Sollya` [11] qui est développée notamment par S. Chevillard et M. Joldeş. `Sollya` est une boîte à outils d'algorithmes numériques d'aide au développement de codes numériques sûrs. Parmi les diverses routines disponibles, elle fournit une norme sup certifiée (que nous avons utilisée dans tout ce chapitre), un évaluateur de fonctions composées en multiprécision et avec arrondi fidèle (voir p.36 pour une définition), une implantation efficace de l'algorithme de Remez qui calcule les meilleures approximations minimax réelles, etc. Elle est utilisée, couplée avec Gappa [12] [45, 47], depuis 2007 pour l'implantation de polynômes d'approximation des fonctions de `CRlibm`.

Au cours des dernières années, S. Chevillard, S. Torres et moi avons amélioré les résultats de **C-4**. Ils sont présentés dans l'article **S-6** que nous rédigeons actuellement.

4.5.2 Déclinaison de notre approche via les réseaux euclidiens

Dans le cadre de l'évaluation de fonctions, deux difficultés supplémentaires se sont présentées à nous. Il y a bien sûr celle de la discrétisation choisie, mentionnée plus haut, qui s'est révélée dépendre du problème visé. Un problème spécifique s'ajoute dans le cadre de l'évaluation de fonction en arithmétique virgule flottante : les coefficients sont des nombres flottants ou des sommes de nombres flottants. Nous allons donc expliquer comment nous procédons au choix des exposants de ces flottants, ce qui nous permet ensuite de nous ramener au problème 4.3

Choix de la discrétisation

C'est une étape critique dans notre approche. En effet, nous voulons qu'une petite valeur de $\|a_0b_0 + \cdots + a_nb_n - t\|_\infty$ puisse impliquer une petite valeur de la norme sup $||p - f||_{[a,b]}$. Plus précisément, exiger que $\|a_0b_0 + \cdots + a_nb_n - t\|_\infty$ soit petit peut être vu comme une sorte de problème d'interpolation approchée. Or, comme nous l'avons vu avec l'exemple de Runge présenté dans la sous-section 3.2, un choix de points hasardeux peut donner lieu à un polynôme certes très proche de la fonction sur la discrétisation choisie mais très éloigné de cette même fonction sur $[a, b]$.

Notre choix de points repose sur l'observation que lorsque la précision imposée sur les coefficients est assez grande (c.-à-d. avec des exposants m_k assez grands, ce qui est le cas avec des coefficients flottants binary64 par exemple), les bons approximants polynomiaux quantifiés de la fonction seront proches de l'approximation minimax. Par conséquent, nous essayons de choisir des points où f et P^*, le polynôme minimax de degré au plus n, sont le plus proches possible, c.-à-d. quand $f - P^*$ s'annule. Le théorème d'équioscillation 4.2 nous assure qu'il existe au moins $n + 1$ tels points.

Les points de Tchebychev de première espèce

$$\mu_k = \frac{a+b}{2} + \frac{b-a}{2}\cos\left(\frac{(k+1/2)\pi}{n+1}\right), k = 0, \ldots, n. \tag{4.5}$$

constituent aussi un choix pertinent. Ils se sont révélés très bons au cours de nos expériences, particulièrement quand la taille des exposants m_k est petite.

Détermination des exposants des coefficients dans le cas flottant

Le problème que nous regardons alors est une généralisation du problème 4.3 :

Problème 4.7 (Approximation minimax en précision finie)**.** *Soit X un ensemble compact de $\mathbb{R}$, $n \in \mathbb{N}$, soient $f \in \mathcal{C}(X)$ et $\{\psi_0, \ldots, \psi_n\}$ une famille d'éléments de $\mathcal{C}(X)$ linéairement indépendants sur $\mathbb{R}$, soit W une fonction poids, déterminer des $a_i, m_i \in \mathbb{Z}, 2^{p-1} \leqslant a_i \leqslant 2^p$ tels que*

$$P = \frac{a_0}{2^{m_0}}\psi_0 + \frac{a_1}{2^{m_1}}\psi_1 + \cdots + \frac{a_n}{2^{m_n}}\psi_n$$

minimise $\|W(P - f)\|_{\infty,X}$.

11. http://sollya.gforge.inria.fr/
12. http://gappa.gforge.inria.fr/

Une idée très simple nous permet d'obtenir des informations rigoureuses sur les coefficients flottants d'une très bonne approximation polynomiale quantifiée. Elle s'appuie sur la programmation linéaire rationnelle [188] et est présentée en détail dans [32]. On se donne une erreur cible ε. Cela peut être une erreur qui garantit d'avoir l'arrondi correct, l'erreur de l'arrondi naïf, etc. On discrétise le problème 4.7 à l'aide de la discrétisation ci-dessus par exemple et l'on obtient un polytope, que l'on projette sur chaque axe à l'aide d'un outil de programmation linéaire. Les intervalles que l'on obtient nous donnent alors beaucoup d'information, à commencer par les binades auxquelles les coefficients doivent appartenir, et donc leurs exposants. Une fois ces derniers fixés, on se retrouve face à une instance du problème 4.3. Serge Torres a réalisé une implémentation de cette approche à l'aide de `GLPK`[13].

Remarque 4.8. Ce pré-calcul se révèle en pratique extrêmement utile puisque nous pouvons en tirer, toujours en se donnant au départ une erreur d'approximation cible, des informations sur :

— le nombre de bits nécessaires à chaque coefficient pour atteindre l'erreur visée. Ainsi, l'outil peut nous indiquer qu'un seul flottant ne suffit pas pour représenter un coefficient et nous dire qu'il vaut mieux travailler avec une somme de deux ou trois flottants par exemple, cf. l'exemple traité dans 4.5.3.

— les valeurs des bits de poids fort (et parfois même les valeurs exactes !) des coefficients d'un polynôme quantifié optimal. Cela a un impact direct sur la qualité de notre méthode. En effet, nous regroupons toutes ces valeurs connues au préalable avec la fonction cible et nous avons alors à résoudre un nouveau problème 4.3 pour lequel notre approche donne en général un résultat plus fin.
Essayons de clarifier cela sur un exemple : on veut approcher une fonction f sur $[1,2]$ à l'aide d'un polynôme $u_0 + u_1 x + u_2 x^2$ de degré au plus 2 et à coefficients u_0, u_1, u_2 dans $\mathcal{F}_{24}$, l'ensemble des flottants en précision 24. On se donne une erreur cible ε qui garantit l'arrondi correct. Le pré-calcul des projections nous indique alors que le coefficient constant vaut $1/2$ et que les exposants respectifs de u_1 et u_2 sont 2 et 0. Notre problème est donc ramené à la détermination de a_1 et $a_2 \in \mathbb{Z}, 2^{23} \leqslant |a_1|, |a_2| \leqslant 2^{24} - 1$ tels que

$$\left\| \frac{a_1}{2^{21}} x + \frac{a_2}{2^{23}} x^2 - \left(f(x) - \frac{1}{2} \right) \right\|_{[1,2]}$$

soit minimal.

Si l'on est à la recherche d'un bon approximant seulement, et que l'on manque de temps pour effectuer ces pré-calculs rigoureux à l'aide de la programmation linéaire, on peut utiliser l'heuristique suivante. Dans ces problèmes d'évaluation de fonction en arithmétique flottante, le format des coefficients donne une souplesse et une grande dynamique à ces nombres, ce qui fait qu'en pratique, un très bon approximant polynomial quantifié P est très proche du minimax P^*. Nous supposons donc que les coefficients de P se trouvent dans la même binade que ceux de P^*, ce qui nous donne des candidats pour les exposants des coefficients flottants de P. Nous sommes donc ramenés à une instance du problème 4.3 et nous exécutons notre approche. Si elle retourne bien des a_k de valeur absolue comprise entre 2^{p-1} et $2^p - 1$, nous sommes satisfaits : nous avons effectivement obtenu des coefficients flottants. Sinon, nous ajustons un peu les exposants m_k et relançons notre code. Nous avons constaté que cette astuce était satisfaisante en pratique après seulement une ou deux étapes [32].

Validation a posteriori

L'idée est d'utiliser de nouveau la programmation linéaire en tenant compte de toutes les informations que nous avons pu collecter auparavant sur les très bons polynômes quantifiés : les valeurs des exposants, les bits de poids fort (ou même les valeurs) des coefficients mais aussi l'erreur d'approximation retournée par notre approche (elle va jouer le rôle de l'erreur cible) ainsi que les extrema de la fonction erreur d'approximation entre la fonction f et le polynôme que nous avons calculé (ils vont jouer le rôle de la discrétisation). On calcule de nouveau un polytope à partir de toutes ces données et on examine les résultats des projections orthogonales sur les axes. En pratique, nous obtenons alors des informations très fines qui nous permettent de garantir que notre solution est très proche de la solution optimale.

13. https://www.gnu.org/software/glpk/

4.5.3 Mise en œuvre pratique

Notre approche est implantée dans la fonction `fpminimax` du logiciel `Sollya`. Nos expérimentations, menées notamment dans le cadre de la synthèse de fonctions pour la bibliothèque `CRlibm`, montrent des résultats extrêmement satisfaisants, que l'on valide à l'aide de nos outils de programmation linéaire, comme nous venons de l'expliquer.

Notre outil permet de traiter deux questions complémentaires :

— on fixe une erreur d'approximation cible et on détermine un polynôme avec un degré et/ou une taille des coefficients aussi petits que possible. Cela est particulièrement intéressant dans les applications matérielles où chaque bit compte.

— On fixe un degré et des formats pour les coefficients et on détermine un polynôme donnant une erreur d'approximation aussi petite que possible.

L'exemple que nous traitons maintenant relève de la seconde catégorie de questions. Il montre le comportement de notre méthode dans une situation concrète. Il provient de l'implémentation de la fonction $\arcsin$ dans la bibliothèque `CRlibm`. Comme on va le voir, il nous offre une situation pratique dans laquelle l'écart entre le polynôme (quasi-)optimal que nous calculons et l'arrondi naïf est très significatif.

Un exemple issu de `CRlibm`

Cet exemple provient de l'implémentation de la fonction $\arcsin$ dans `CRlibm`. Rappelons qu'il s'agit d'une bibliothèque mathématique qui garantit l'évaluation avec arrondi correct en format binary64. Pour obtenir l'arrondi correct, les développeurs de `CRlibm` doivent utiliser une précision plus grande que la précision binary64 (ou double précision) fournie par le processeur. À cet effet, ils se servent de formats de coefficients spéciaux tels que le double-`double` et le triple-`double` qui sont des sommes, non évaluées, de deux, respectivement trois, flottants binary64. En particulier, ils approchent les fonctions à l'aide de polynômes dont les coefficients peuvent être des flottants binary64, des double-`double` ou des triple-`double`.

On va maintenant construire une approximation d'$\arcsin$ sur l'intervalle $[0.79; 1]$. C'est un exemple développé avec C. Lauter (un des principaux développeurs de `CRlibm`). Après une réduction d'argument, nous nous retrouvons à devoir approcher

$$g(z) = \frac{\arcsin(1-(z+m)) - \frac{\pi}{2}}{\sqrt{2\cdot(z+m)}}$$

où $-0.110 \lesssim z \lesssim 0.110$ et $m \simeq 0.110$. Les développeurs savent grâce aux travaux de Lefèvre et al. [125, 126, 127] qu'ils ont besoin d'une précision de 119 bits pour assurer l'arrondi correct. L'approximation minimax de degré 21 fournit une telle précision puisque l'erreur d'approximation associée vaut $2^{-119.83}$.

Notre étude préalable, à l'aide de programmation linéaire, cf. remarque 4.8, nous indique que le polynôme que nous recherchons est de la forme

$$\underbrace{p_0}_{t.d.} + \underbrace{p_1}_{t.d.} x + \underbrace{p_2}_{d.d.} x^2 + \underbrace{\cdots}_{\cdots} + \underbrace{p_9}_{d.d.} x^9 + \underbrace{p_{10}}_{d.} x^{10} + \underbrace{\cdots}_{\cdots} + \underbrace{p_{21}}_{d.} x^{21}$$

où les p_i sont soit des flottants binary64 (d.), soit la somme de deux flottants binary64 (d.d.), soit la somme de trois flottants binary64 (t.d.). Elle est même, en fait, plus précise que cela puisque nous pouvons obtenir la valeur du premier flottant de chaque double-`double` et des deux premiers flottants de chaque triple-`double` de ce polynôme d'approximation.

Notre méthode renvoie instantanément un polynôme avec deux triple-`doubles`, huit double-`doubles` et douze flottants binary64.

Comme le montre la table 4.1, la précision que donne notre polynôme est très proche de celle du minimax. L'utilisation de l'arrondi naïf n'apporterait que 103 bits de précision, ce qui serait insuffisant pour garantir l'arrondi correct. Cet exemple illustre bien l'écart significatif que l'on peut rencontrer parfois entre l'arrondi naïf et le polynôme quantifié optimal. Ici, l'utilisation de l'arrondi naïf ferait perdre 16 précieux bits. La comparaison de la qualité d'approximation se voit très bien sur les figures 4.1 et 4.2.

TABLE 4.1 – Logarithme en base 2 de l'erreur absolue d'approximation

Cible	-119
Minimax	-119.83
Minimax arrondi	-103.31
Notre approche	-119.77

En particulier, si l'on se rappelle le théorème 4.2 qui nous indique qu'un polynôme est optimal si et seulement si l'erreur d'approximation équioscille en $n+2$ de ses extrema, on voit que notre polynôme est proche de satisfaire cette dernière condition. On peut même s'aider du théorème 3.14 de La Vallée Poussin pour quantifier très précisément cette observation.

FIGURE 4.1 – Erreur minimax et erreur entre g et l'arrondi naïf

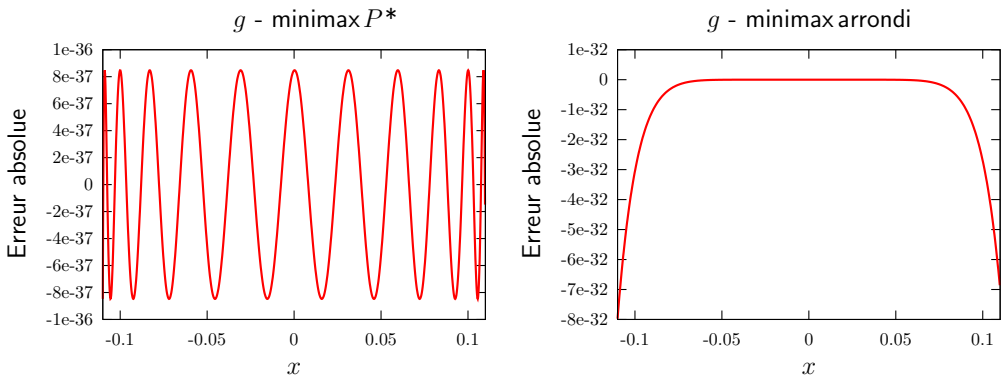
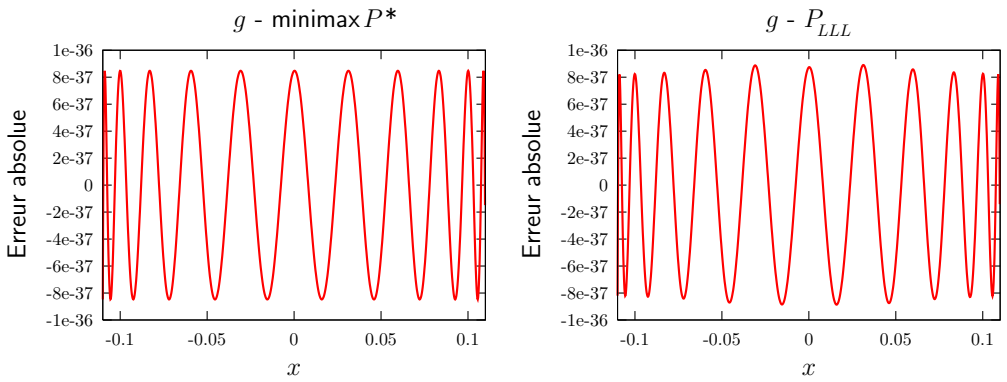


FIGURE 4.2 – Erreur minimax et erreur entre g et notre polynôme P_{LLL}



4.5.4 Approximation polynomiale avec coefficients contraints : le cas de la norme L^2

Dans **C-5**, G. Hanrot et moi-même nous sommes attaqués au problème 4.3, considéré cette fois dans le cadre L^2. Nous nous sommes focalisés sur le cas de l'approximation polynomiale à coefficients flottants

mais il faut préciser que la méthode que nous avons mise au point est très souple et que l'on peut considérer de nombreuses autres contraintes sur les coefficients (virgule fixe, fixation des valeurs de certains coefficients) et d'autres familles génératrices que les monômes $\{1, \ldots, x^n\}$.

Comme nous l'avons rappelé au début de ce chapitre, la détermination de la meilleure approximation polynomiale au sens L^2 à coefficients réels est un exercice facile de projection orthogonale. Mais, ici encore, lorsque nous exigeons des coefficients qu'ils soient en précision finie, arrondir au plus près les coefficients de la meilleure approximation à coefficients réels peut dégrader significativement la situation et la détermination d'une approximation optimale devient nettement plus difficile. Nous avons pu montrer dans notre travail que ce problème est en fait NP-difficile, par réduction à un CVP.

Malgré cela, la méthode que nous avons mise au point permet, pour les tailles des degrés et coefficients usuellement mobilisés dans le cadre de l'approximation de fonction, d'obtenir rapidement une meilleure approximation L^2 quantifiée.

4.6 Synthèse de filtres RIF

Comme je l'ai mentionné plus haut, je réfléchis depuis quelques années à des questions analogues dans le cadre du traitement de signal. Au cœur de cette réflexion se trouve le doctorat de Silviu-Ioan Filip, que j'ai co-encadré avec G. Hanrot et dont Silviu vient de soutenir la thèse. Silviu a travaillé sur des questions d'approximation et leur application à la synthèse de filtres à réponse impulsionnelle finie (RIF) et à réponse impulsionnelle infinie (RII) efficaces en machine. Il a développé depuis son stage de M2 une implantation robuste de l'algorithme de Parks-McClellan [67], qui est au cœur de la synthèse de filtres RIF. Son implantation est maintenant plus rapide que les concurrentes et surtout plus robuste au sens où elle continue à converger dans des cas extrêmes qui sont mal traités ou inaccessibles aux autres implantations. Nous avons aussi avancé substantiellement sur la question de la quantification de ces filtres. Outre son intérêt pratique évident, la question s'est révélée très intéressante pour nous car la méthode s'appuyant sur les réseaux euclidiens que nous avons utilisée avec succès pour l'approximation de fonctions élémentaires ne s'adapte pas directement et cela a soulevé plusieurs questions qui nous ont permis d'affiner notre compréhension du problème. Silviu, G. Hanrot et moi-même avons rédigé et soumis un article de journal sur cette question des fonctions de transfert efficaces en machine **S-3**.

Enfin, Silviu, F. de Dinechin, M. Istoan (INSA Lyon) et moi-même nous sommes appuyés notamment sur ce travail pour développer un outil intégré en C++ permettant la synthèse automatique de filtres RIF sur circuits reconfigurables FPGAs. Partant des spécifications du filtre, l'outil renvoie du code VHDL optimisé d'évaluation du filtre. Ce travail est décrit dans **S-4**.

4.6.1 Introduction

La synthèse efficace de filtres RIF est un sujet de recherche actif en traitement du signal depuis plusieurs décennies. À l'instar de ce que nous avons montré pour l'évaluation de fonctions, la nature discrète des coefficients (en général, en virgule fixe ou virgule flottante) du filtre implémenté entraîne des difficultés importantes pour leur détermination.

Ici, les contraintes usuelles sont d'exiger que les coefficients soient représentables sur des entiers de b bits signés (à multiplication près par un facteur d'échelle) ou encore qu'ils aient une petite complexité (on va privilégier des entiers qui sont sommes d'un petit nombre de puissances de 2). Le premier type de contraintes est utile pour l'implémentation sur processeurs de signal numérique (DSP en anglais) en virgule fixe. Le second type de contraintes est quant à lui relié à la construction de circuits de filtrage sans multiplieurs. Ce dernier problème est abordé dans [131, 134, 133, 132, 135, 220, 192, 27, 42] par exemple. Nous allons nous focaliser ici sur la première question, dans le cas des filtres RIF en forme directe. Ce problème de quantification des coefficients a été abordé dès les années 1960 [109] (on peut consulter [110] dont l'introduction donne un compte-rendu des travaux liés à cette question dans les années 1970).

En général, déterminer un filtre FIR, sous forme directe, à coefficients en précision finie optimal se révèle très coûteux au fur et à mesure que le degré s'élève. Les outils de résolution exacte utilisent des techniques PLEM, combinées parfois avec des stratégies de séparation et évaluation (branch-and-bound en anglais) astucieuses [131, 110, 115, 134, 133, 135, 81, 191, 112, 113]. Pour traiter les problèmes de degré

grand (pour fixer les idées : à partir du degré 35 disons), des heuristiques plus rapides ont été proposées. Elles fournissent des résultats qui sont dans de nombreux cas quasi-optimaux et peuvent aider à accélérer les outils de résolution exacts. L'approche présentée dans [165] est utilisée au sein de MATLAB™ pour traiter ce problème de quantification des coefficients de filtres FIR. Les méthodes développées dans [132] (voir aussi [63] pour une variante) et [114] produisent aussi de très bons résultats.

Comme nous allons le voir dans la section suivante, la formulation de ce problème de quantification de filtres FIR est très proche de celle du problème de détermination de bons polynômes d'approximation contraints que nous venons d'étudier. Ce qui est intéressant, c'est que le fait que les coefficients soient ici plus contraints encore empêche une adaptation directe de la méthode utilisée pour les polynômes. Cela nous a amenés à affiner notre outil de résolution approchée de CVP_2, en utilisant cette fois-ci le plongement de Kannan et surtout à étudier plus finement le problème de la discrétisation, pour lequel nous proposons trois familles distinctes et complémentaires.

Notre approche se révèle robuste, particulièrement quand on s'attaque à des synthèses de grand degré. Silviu a réalisé une implémentation en C++ open source de notre méthode [14]. Elle retourne des filtres (quasi-)optimaux et doit permettre l'accélération des outils de résolution exacte.

4.6.2 Quantification des filtres FIR sous forme directe

Nous rappelons [15] maintenant des notions classiques sur la synthèse de filtres RIF à phase linéaire [168, 6, 176]. Pour ce faire, on se place généralement dans le domaine fréquentiel. Dans le cas d'un filtre causal à N coefficients à réponse impulsionnelle réelle, la réponse fréquentielle que l'on souhaite optimiser est

$$H(\omega) = \sum_{k=0}^{N-1} h[k]e^{-i\omega k} = G(\omega)e^{i\left(\frac{L\pi}{2} - \frac{N-1}{2}\omega\right)},$$

où G est une fonction à valeurs réelles et L vaut 0 ou 1. Traditionnellement, il y a quatre types pour ce genre de filtres que l'on considère en pratique, numérotés de I à IV ; ils dépendent de la parité de la longueur du filtre N et de la symétrie des $h[k]$ par rapport aux coefficients médians. On peut écrire $G(\omega)$ sous la forme $Q(\omega)P(\omega)$, où $P(\omega)$ est de la forme :

$$P(\omega) = \sum_{k=0}^{n} p_k \cos(k\omega).$$

Suivant le type de filtre, $Q(\omega)$ vaut $1, \cos(\omega/2), \sin(\omega)$ ou $\sin(\omega/2)$. Nous avons $n = \lfloor N/2 \rfloor$ et des formules simples lient les $h[k]$ aux p_k (voir par exemple [6, Ch. 15.8–15.10]).

Remarque 4.9. Si l'on considère le changement de variable $x = \cos(\omega)$, $P(\omega) = \sum_{k=0}^{n} p_k \cos(k\omega)$ est en fait un polynôme de degré au plus n en $\cos(\omega)$ exprimé dans la base des polynômes de Tchebychev de première espèce $(T_k)_{0 \leqslant k \leqslant n}$, c.-à-d.

$$P(\omega) = \sum_{k=0}^{n} p_k T_k(\cos(\omega)) = \sum_{k=0}^{n} p_k T_k(x).$$

Quand on étudie ces questions, il est parfois commode de se placer plutôt dans le cadre de l'approximation polynomiale. Quand c'est le cas, x est alors la variable d'approximation et $X \subseteq [-1, 1]$ le domaine associé à Ω, c.-à-d. $\cos(\Omega)$.

Dans ce qui suit, nous allons nous concentrer sur les filtres de type I (à savoir $Q \equiv 1$), sachant que l'adaptation aux trois autres cas est immédiate. Les réponses fréquentielles que nous manipulons sont donc de la forme

$$H(\omega) = \sum_{k=0}^{n} p_k \cos(k\omega). \tag{4.6}$$

La réponse fréquentielle, en précision infinie, de longueur N optimale est la solution du problème 4.1, qui se formule ainsi dans ce cadre :

14. Voir https://github.com/sfilip/fquantizer.
15. Une partie de ce passage suit la présentation donnée dans la Section II de [113].

TABLE 4.2 – Spécifications des filtres considérés dans [114].

filtre	bandes	$D(\omega)$	$W(\omega)$
A	$[0, 0.4\pi]$	1	1
	$[0.5\pi, \pi]$	0	1
B	$[0, 0.4\pi]$	1	1
	$[0.5\pi, \pi]$	0	10
C	$[0, 0.24\pi]$	1	1
	$[0.4\pi, 0.68\pi]$	0	1
	$[0.84\pi, \pi]$	1	1
D	$[0, 0.24\pi]$	1	1
	$[0.4\pi, 0.68\pi]$	0	10
	$[0.84\pi, \pi]$	1	1
E	$[0.02\pi, 0.42\pi]$	1	1
	$[0.52\pi, 0.98\pi]$	0	1

Problème 4.10 (Approximation minimax de filtre RIF)**.** *Soit Ω un compact de $[0, \pi]$, D une réponse fréquentielle idéale continue sur Ω, soit un poids W défini sur Ω. Pour un degré du filtre $n \in \mathbb{N}$, on veut déterminer $P^*(\omega) = \sum_{k=0}^{n} p_k^* \cos(\omega k)$ tel que l'erreur pondérée $E^*(\omega) = W(\omega)\,(P^*(\omega) - D(\omega))$ ait une norme sup*

$$\|E^*\|_{\infty,\Omega} = \sup_{\omega\in\Omega} |E^*(\omega)|\,,$$

minimale.

Pour donner une idée un peu plus précise des objets que nous considérons, regardons par exemple un ensemble de spécifications de filtres de type I donnée dans [114, Table 1], que nous reproduisons dans la table 4.2.

La manière usuelle de résoudre le problème 4.1 est de faire appel, comme nous l'avons mentionné dans la section 4.6.1, à la variante de l'algorithme de Remez due à Parks et McClellan.

La déclinaison du problème 4.3 ici est la quantification des coefficients de la réponse fréquentielle (4.6) en virgule fixe. Nous utilisons des nombres virgule fixe sur b bits, sous la forme m/s, où m est un entier dans $I_b = \{-2^{b-1}, \ldots, -1, 0, 1, \ldots, 2^{b-1} - 1\}$ et s est un facteur d'échelle fixé qui, dans de nombreux cas, est une puissance de 2. Soit $\varphi_0(\omega) = 1/s$ et $\varphi_k(\omega) = 2\cos(k\omega)/s$ pour $k = 1, \ldots, n$.

Problème 4.11 (Quantification des coefficients d'un filtre RIF)**.** *Soit Ω un compact de $[0, \pi]$, D une réponse fréquentielle idéale continue sur Ω, soit un poids W défini sur Ω. Pour un entier n donné, déterminer $P(\omega) = \sum_{k=0}^{n} a_k \varphi_k(\omega)$, où $a_k \in I_b$ pour $k = 0, \ldots, n$, tel que*

$$\|W\,(P - D)\|_{\infty,\Omega} \tag{4.7}$$

soit minimale.

Comme déjà mentionné plus haut, ce problème a toujours une solution et elle n'est pas nécessairement unique.

Si l'on voit ce problème sous l'angle donné par la formulation 4.1, Ω est en général remplacé par une discrétisation finie Ω_d, donnant ainsi lieu à une instance de PLEM. Un nombre de points égal à $16n$ suffit habituellement [111].

Le facteur d'échelle s, interprété en général comme le gain du filtre, peut aussi être un paramètre de la synthèse. L'optimisation de s peut aussi améliorer la qualité de l'erreur de quantification (4.7) par un petit facteur. Toutefois, trouver le meilleur s possible semble non-trivial [130, 111, 112].

Notre approche permet aussi de considérer des problèmes de quantification plus généraux, où les coefficients ont des formats distincts (en virgule fixe ou en virgule flottante). Ils peuvent se formuler dans le cadre du problème 4.3, chaque coefficient possédant son propre facteur d'échelle $s_k, k = 0, \ldots, n$ sous la forme d'une puissance de 2. Néanmoins, afin de simplifier notre exposé, nous ne considérons que des exemples de quantification relevant du problème 4.11.

4.6.3 Discrétisation du problème

Le choix des ω_i (ou $x_i = \cos(\omega_i)$) est critique pour l'obtention de bons résultats. Nous avons vu que dans le cadre de l'évaluation de fonctions élémentaires sur un intervalle compact $X = [a, b]$, les choix de prédilection sont les zéros de E^* ou les nœuds de Tchebychev (4.5) de première espèce d'ordre n. Cependant, dans le cadre de la synthèse de filtre, Ω (et par conséquent $X = \cos \Omega$) est l'union d'au moins deux intervalles, et on ne dispose pas à ce jour d'expressions sous forme close de fréquences ω_i (et de points x_i) adéquates. Nous avons donc été amenés à proposer trois alternatives complémentaires qui se sont révélées performantes au cours de nos expérimentations. L'idée sous-jacente aux deux premiers choix est de forcer la fonction de transfert en précision finie à imiter l'approximation minimax P^*, tandis que le troisième correspond à un choix judicieux d'initialisation de l'algorithme de Parks-McClellan déterminant P^*.

« Zéros » de la fonction d'erreur minimax

Cette stratégie est l'analogue direct du choix présenté dans la section 4.5.2. Toutefois, il y a une différence significative : dans le cas d'un intervalle compact $[a, b]$, le théorème 4.2 et le théorème des valeurs intermédiaires garantissent qu'il existe au moins $n + 1$ points $x_0, \ldots, x_n$ dans $[a, b]$ tel que $E^*(x_i) = 0$ pour $i = 0, \ldots, n$. Malheureusement, nous n'avons aucune telle garantie dans le cas de l'union de plusieurs intervalles : le nombre de zéros peut être inférieur à $n + 1$. Nous complétons donc cette liste en prenant des points situés au milieu des extrémités de deux bandes consécutives. Ces points n'appartiennent pas à Ω mais c'est un problème facile à régler, cf. **S-3**.

Remarque 4.12. Il n'est pas difficile de démontrer que si un filtre a p bandes alors E^* a au moins $n + 2 - p$ zéros et au plus $n + p - 1$ zéros sur ces bandes (avec $p \geqslant 2$). Donc, si il y a $p - 1$ points qui sont au milieu des extrémités de deux bandes consécutives, nous avons la garantie que la discrétisation recherchée a au moins $n + 1$ points et au plus $n + p - 1$ points. Dans les cas pratiques de la section 4.6.4, cette discrétisation a exactement $n + 1$ points (filtre à 2 bandes) ou au plus $n + 2$ points (filtre à 3 bandes).

Extrema équioscillants de la fonction d'erreur minimax

Inspirée de la caractérisation que l'on trouve dans le théorème 4.2, la discrétisation Ω_e suggérée est un ensemble de $n + 2$ fréquences qui donnent $n + 2$ extrema équioscillant pour la fonction d'erreur E^*.

Il y a des cas où le nombre de fréquences donnant lieu à des extrema équioscillant est plus grand que $n + 2$. Il nous faut donc un procédé efficace pour sélectionner exactement $n + 2$ fréquences : l'algorithme de Parks-McClellan construit itérativement cette liste.

Remarque 4.13. En pratique, nous utilisons l'implémentation de l'algorithme de Parks-McClellan élaborée dans [67] et disponible depuis `https://github.com/sfilip/firpm` : elle calcule P^* (ou plutôt une approximation de P^* aussi fine que souhaitée) et une liste de $n + 2$ fréquences où E^* équioscille, qui est donc notre Ω_e.

Points de Fekete approchés

Silviu Filip a eu l'idée d'utiliser cette approche pour obtenir un procédé numériquement robuste d'initialisation de l'algorithme de Parks-McClellan [67, Sec. 4.1–4.2]. Nous la rappelons brièvement.

Si $X \subset \mathbb{R}$ est compact, on définit les points de Fekete de degré $n + 2$ comme les éléments d'un ensemble $\{t_0, \ldots, t_{n+1}\} \subset X$ qui maximisent la valeur absolue du déterminant de type Vandermonde $|w(y_i)T_j(y_i)|_{0 \leqslant i,j \leqslant n+1}$, avec $w(x) = W(\arccos(x))$, où W est la fonction poids utilisée dans les énoncés des problèmes 4.1 and 4.3 et les T_j sont les $n + 2$ polynômes de Tchebychev de première espèce.

Malheureusement, leur détermination est un problème calculatoire difficile. Toutefois, des approximations de ces points, que l'on appelle points de Fekete approchés (PFA) peuvent être déterminées efficacement. L'idée est de remplacer X par un sous-ensemble bien choisi $Y \subseteq X$ à $m + 1 (\geqslant n + 2)$ éléments et d'extraire ensuite des points de Fekete de Y. Un choix pertinent est de considérer ce que l'on

appelle des mailles faiblement admissibles (WAM pour weakly admissible meshes en anglais) [26]. Ici, la WAM Y que nous considérons est l'union des familles de nœuds de Tchebychev de seconde espèce

$$\nu_k = \frac{a+b}{2} + \frac{b-a}{2}\cos\left(\frac{k\pi}{n+1}\right), k = 0, \ldots, n+1,$$

adaptés à chaque intervalle composant X. Même dans ce cas particulier, la détermination des points de Fekete sur Y est un problème NP-difficile [29]. On peut toutefois utiliser un algorithme glouton efficace qui s'appuie sur l'algorithme QR avec pivotage de colonne [22, 21, 193, 194] :

Algorithme 10 Calcul de points de Fekete approchés

Entrée : Un ensemble fini $Y = \{y_0, \ldots, y_m\} \subseteq X$

Sortie : Une discrétisation à $n+2$ éléments $X_{\text{pfa}} \subset Y$ telle que la sous-matrice de type Vandermonde $\mathbf{V}(X_{\text{pfa}})$ engendrée par les éléments de X_{pfa} a un grand volume

// Initialisation. $\mathbf{V}(Y)$ est la matrice de type Vandermonde

// $(w(y_i)T_j(y_i))_{0\leqslant i\leqslant m, 0\leqslant j\leqslant n+1}$

1: $\mathbf{A} \leftarrow \mathbf{V}(Y), b \in \mathbb{R}^{n+2}, b \leftarrow (1 \cdots 1)^t$

// Résolution du système linéaire à l'aide de QR

2: En utilisant une routine QR avec pivotage de colonne (via un analogue de la routine DGEQP3 de LAPACK [5]), calculer $\mathbf{w} \in \mathbb{R}^{m+1}$, une solution au système sous-déterminé $\mathbf{b} = \mathbf{A}^t\mathbf{w}$.

// Sélection du sous-ensemble

3: On choisit pour X_{pfa} l'ensemble des éléments de Y dont les termes correspondant au sein de $\mathbf{w}$ sont non nuls, c'est-à-dire, si $y_i \in Y$ et $w_i \neq 0$, alors $y_i \in X_{\text{pfa}}$.

En pratique, on travaille avec $\Omega_{\text{afp}} = \arccos(X_{\text{pfa}})$.

4.6.4 Résultats expérimentaux

Pour donner une idée de la qualité de notre méthode, nous la comparons à l'approche de l'arrondi télescopique introduite dans [114]. Les tests que nous présentons ont été effectués sur un Intel i7-3687U CPU avec un système Linux 64-bit et un compilateur `g++` version 5.3.0 C++. Les trois approches de discrétisation de la section 4.6.3 sont utilisées, en ne conservant que les meilleurs résultats obtenus.

Arrondi télescopique de Kodek et Krisper [114]

Nous allons éprouver notre code sur une banque de spécifications de filtres de type I donnée dans [114, Table 1], reproduite dans la table 4.2. À partir de ces spécifications, on forme un ensemble de filtres pratiques présentés dans la première colonne de la table 4.2. Par exemple, A35/8 désigne un problème de synthèse suivant la spécification A, de longueur $N = 35$ (ce qui signifie un degré d'approximation au plus $n = 17$, $b = 8$ bits utilisés pour stocker les coefficients du filtre (bit de signe inclus) et un facteur d'échelle $s = 2^{b-1} = 2^7$.

Les résultats sont présentés dans la table 4.3 :

— la première colonne liste les spécifications ;

— la deuxième donne l'erreur minimax E^* calculée à l'aide de l'algorithme de Parks-McClellan.

Toutes les colonnes restantes présentent les erreurs d'approximation des fonctions de transfert obtenues à l'aide de diverses stratégies de quantifications :

— la troisième colonne présente des erreurs de quantification E_{opt} optimales (ou les meilleures connues à ce jour) : les erreurs pour les filtres de longueur 125 ne sont pas prouvées optimales contrairement aux dix autres erreurs. Si l'on excepte les filtres signalés par ‡, ils sont obtenus en utilisant un outil de PLEM (les valeurs sont celles de [114, Table 2]). Dans le cas des problèmes signalés par †, précisons que l'outil PLEM exact est arrêté au bout d'un certain temps (ici une vingtaine de minutes). Nous indiquons aussi les méthodes qui permettent d'atteindre ces erreurs, avec M = PLEM (limité en temps), L = notre approche, T = arrondi télescopique à deux coefficients [114, Sec. 4].

TABLE 4.3 – Comparaison des erreurs de quantifications pour les spécifications de filtre données dans [114].

Filtre	Minimax E^*	Meilleure erreur quantifiée connue E_{opt}	Arrondi naïf $E_{\text{naïve}}$	Télescopage $E_{\text{tél}}$	Réduction LLL E_{LLL}	Réduction BKZ E_{BKZ}	Réduction HKZ E_{HKZ}
A35/8	0.01595	0.02983 (M,L)	0.03266	0.03266	**0.02983**	**0.02983**	**0.02983**
A45/8	$7.132 \cdot 10^{-3}$	0.02962 (M,L)	0.03701	0.03186	**0.02962**	**0.02962**	**0.02962**
A125/21†	$8.055 \cdot 10^{-6}$	$1.077 \cdot 10^{-5}$ (M)	$1.620 \cdot 10^{-5}$	$1.179 \cdot 10^{-5}$	$\mathbf{1.161 \cdot 10^{-5}}$	$1.168 \cdot 10^{-5}$	$1.251 \cdot 10^{-5}$
B35/9	0.05275	0.07709 (M)	0.15879	**0.07854**	0.08205	0.08205	0.08205
B45/9	0.02111	0.05679 (M)	0.11719	0.06641	**0.06041**	**0.06041**	**0.06041**
B125/22†	$2.499 \cdot 10^{-5}$	$2.959 \cdot 10^{-5}$ (M)	$6.198 \cdot 10^{-5}$	$3.293 \cdot 10^{-5}$	$\mathbf{3.243 \cdot 10^{-5}}$	$3.344 \cdot 10^{-5}$	$3.344 \cdot 10^{-5}$
C35/8	$2.631 \cdot 10^{-3}$	0.01787 (M,T,L)	0.04687	**0.01787**	**0.01787**	0.01917	0.01917
C45/8	$6.709 \cdot 10^{-4}$	0.01609 (M,L)	0.03046	0.02103	**0.01609**	**0.01609**	0.02291
C125/21‡	$1.278 \cdot 10^{-8}$	$1.564 \cdot 10^{-6}$ (L)	$8.203 \cdot 10^{-6}$	$2.1 \cdot 10^{-6}$	$1.606 \cdot 10^{-6}$	$\mathbf{1.564 \cdot 10^{-6}}$	$\mathbf{1.564 \cdot 10^{-6}}$
D35/9	0.01044	0.03252 (M,L)	0.12189	0.03368	**0.03252**	0.03291	0.03291
D45/9	$2.239 \cdot 10^{-3}$	0.02612 (M)	0.10898	0.02859	**0.02706**	0.02805	**0.02706**
D125/22‡	$4.142 \cdot 10^{-8}$	$1.781 \cdot 10^{-6}$ (L)	$3.425 \cdot 10^{-5}$	$2.16 \cdot 10^{-6}$	$1.864 \cdot 10^{-6}$	$\mathbf{1.781 \cdot 10^{-6}}$	$1.826 \cdot 10^{-6}$
E35/8	0.01761	0.03299 (M)	0.04692	0.03404	**0.03349**	**0.03349**	**0.03349**
E45/8	$6.543 \cdot 10^{-3}$	0.02887 (M,L)	0.03571	0.03403	0.03167	0.03094	**0.02887**
E125/21†	$7.889 \cdot 10^{-6}$	$1.034 \cdot 10^{-5}$ (M)	$1.479 \cdot 10^{-5}$	$\mathbf{1.127 \cdot 10^{-5}}$	$1.215 \cdot 10^{-5}$	$1.245 \cdot 10^{-5}$	$1.162 \cdot 10^{-5}$

— la colonne quatre liste les erreurs obtenues en arrondissant simplement les coefficients du minimax en précision infinie vers les valeurs les plus proches satisfaisant au format de quantification imposé (arrondi naïf) ;
— la cinquième colonne donne les erreurs que l'on obtient en utilisant l'approche d'arrondi télescopique à deux coefficients de [114, Sec. 4] ;
— les trois dernières colonnes montrent les erreurs de quantification que l'on obtient en utilisant comme algorithme de réduction de réseau LLL, BKZ et HKZ respectivement, lorsqu'on applique l'algorithme 9. Nous avons utilisé une taille de bloc de 8 lors de l'appel à BKZ.

Pour les quatre dernières colonnes, les valeurs données en gras sont les meilleures calculées à l'aide des approches heuristiques.

On remarque que notre approche à base de réseaux euclidiens fournit des résultats optimaux (ou les meilleurs connus) dans huit cas sur quinze, les sept autres cas présentant des résultats très proches des résultats optimaux. Notre approche est supérieure à l'arrondi télescopique dans douze cas sur quinze et donne la même valeur (optimale) dans un treizième cas : l'utilisation de l'option LLL fournit un meilleur résultat dans douze cases et un résultat (optimal) identique dans un treizième cas, l'utilisation de l'option BKZ donne une résultat meilleur dans onze cas et celle de HKZ dans neuf cas. Remarquons, en particulier, le bon comportement de l'algorithme LLL dans les quinze cas, ce qui souligne une spécificité des réseaux euclidiens que nous réduisons : leurs bases en sortie sont d'excellente qualité, supérieure à ce que l'on peut attendre de LLL en pratique, cf. section 4.6.5. Notre approche semble fonctionner particulièrement bien quand l'écart entre l'erreur minimax et l'erreur de l'arrondi naïf est significative, ce qui constitue bien sûr la cas le plus pertinent pour notre étude. Enfin, nous notons que dans les cas C125/21 et D125/22, notre approche retourne (et cela en moins de 8 secondes, comme nous allons maintenant le voir) des résultats meilleurs que ceux retournées par les outils PLEM (limités en temps).

Temps de calcul en pratique

L'analyse du temps d'exécution de l'algorithme 9 dépend de l'algorithme de réduction de réseau utilisé pour le plongement de Kannan (lignes 6 et 7), cf. les sous-sections 2.6.1 et 2.6.2, et de la recherche au voisinage (lignes 8 et 9), qui prend un temps $O(n^3)$. En fait, les bases des réseaux qui surviennent dans nos problèmes sont, au départ, presque déjà réduites, ce qui rend la portion dédiée au plongement de Kannan bien plus rapide que la recherche au voisinage (au moins lorsqu'on utilise les réductions LLL et BKZ). On peut le constater par exemple dans la table 4.4, qui détaille le temps d'exécution de notre code, en utilisant les trois discrétisations de la section 4.6.3, sur les exemples de la table 4.3 :

— la première colonne liste les spécifications ;
— la deuxième colonne donne le temps de calcul total pour les trois discrétisations ;
— les trois colonnes suivantes présentent le temps de calcul dévolu à la partie du code consacrée à la

TABLE 4.4 – Temps d'exécution (en secondes) lorsqu'on utilise les trois discrétisations de la section 4.6.3. La dernière colonne donne le temps de calcul total lorsque LLL est choisi comme algorithme de réduction de réseau.

Filtre	Calcul de la discrétisation	LLL	BKZ	HKZ	Recherche au voisinage	Total (LLL)
A35/8	0.048	0.026	0.028	0.028	0.231	0.325
A45/8	0.047	0.033	0.035	0.035	0.474	0.571
A125/21	0.112	0.291	0.348	0.471	7.686	8.185
B35/9	0.112	0.018	0.019	0.027	0.243	0.401
B45/9	0.102	0.033	0.035	0.045	0.468	0.676
B125/22	0.191	0.296	0.342	1.194	7.746	8.314
C35/8	0.094	0.022	0.021	0.021	0.225	0.396
C45/8	0.107	0.032	0.036	0.037	0.414	0.611
C125/21	0.319	0.455	0.671	303.98	6.366	7.278
D35/9	0.162	0.019	0.021	0.021	0.211	0.411
D45/9	0.096	0.031	0.037	0.037	0.411	0.595
D125/22	0.251	0.435	0.784	335.38	6.240	7.042
E35/8	0.063	0.017	0.018	0.021	0.211	0.301
E45/8	0.089	0.018	0.031	0.031	0.420	0.574
E125/21	0.236	0.261	0.312	1.935	1.184	7.712

TABLE 4.5 – Spécification de filtres FIR de type I de haut degré

Filtre	Bandes	$D(\omega)$	$W(\omega)$
F	$[0, 0.2\pi]$	0	10
	$[0.205\pi, \pi]$	1	1
G	$[0, 0.4\pi]$	1	1
	$[0.405\pi, 0.7\pi]$	0	10
	$[0.705\pi, \pi]$	1	1
H	$[0, 0.45\pi]$	0	10
	$[0.47\pi, \pi]$	1	1

réduction de réseaux et la résolution approchée du CVP_2, ce qui correspond aux lignes 6 et 7 de l'algorithme 9) lorsque l'on choisit les options LLL, BKZ et HKZ respectivement ;

— la sixième colonne fournit le temps de calcul de la recherche au voisinage, qui correspond aux lignes 8 et 9 de l'algorithme 9 ;

— la dernière colonne donne le temps d'exécution total de notre code lorsque l'on choisit l'option LLL. Les étapes restantes sont exécutées très rapidement, si bien que les valeurs dans cette colonne sont à peine plus grandes que la somme des valeurs présentées dans les colonnes 2, 3 et 6).

Nous avons utilisé les implémentations des algorithmes LLL, BKZ et HKZ de la bibliothèque C++ `fplll` [202].

On constate que notre approche est rapide lorsque l'on choisit l'option LLL : par exemple, il faut au plus 8 secondes pour calculer une très bonne réponse fréquentielle pour les filtres de longueur 125. Fait intéressant, l'utilisation de l'option BKZ, qui calcule des bases plus réduites, est à peine plus coûteuse en temps. Cette option BKZ nous permet de calculer les meilleures réponses fréquentielles connues à ce jour pour les filtres C125 et D125 en moins de 8 secondes. Un autre fait remarquable est l'excellent comportement de notre approche quand on choisit l'option HKZ : notre code est très rapide dans les cas des filtres A125, B125 et E125 et donne un temps d'exécution raisonnable dans les cas C125 et D125 où les réseaux correspondants ont un rang égal à 63, ce qui est habituellement une dimension très difficile à traiter pour HKZ.

Remarque 4.14. Pour illustrer plus avant sa robustesse, nous mentionnons que nos routines passent bien à l'échelle, en permettant de s'attaquer à des problèmes de bien plus grand degré encore, comme on le

TABLE 4.6 – Résultats de quantification en grand degré

Filtre	Minimax $E^\star$	Arrondi naïf $E_{\text{naïve}}$	Réduction LLL E_{LLL}	Temps d'exécution en secondes
F1601/18	$8.64 \cdot 10^{-4}$	$3.41 \cdot 10^{-3}$	$\mathbf{1.20 \cdot 10^{-3}}$	846.85
G1201/16	$4.78 \cdot 10^{-3}$	$1.21 \cdot 10^{-2}$	$\mathbf{5.79 \cdot 10^{-3}}$	567.01
H501/16	$1.67 \cdot 10^{-4}$	$5.79 \cdot 10^{-3}$	$\mathbf{1.44 \cdot 10^{-3}}$	379.11

voit dans la table 4.6, qui correspond aux spécifications de la table 4.5. Le facteur d'échelle utilisé ici est encore $s = 2^{b-1}$.

4.6.5 Pourquoi cela fonctionne t-il si bien ?

Comme je l'ai souligné plus haut, notre approche donne en pratique d'excellents résultats dans le cas de l'évaluation de fonction et des résultats oscillant entre très bons et excellents (et qui améliorent l'existant) pour la synthèse de filtres RIF. Cela peut paraître surprenant si l'on considère la suite, un peu acrobatique à première vue, de simplifications ou d'approximations que nous opérons : choix d'une discrétisation aussi parcimonieuse que possible, substitution d'un CVP_2 au CVP_∞ qui résultait de la discrétisation et pour finir, résolution (seulement) approchée de ce CVP_2.

Discrétisation

Ce point est, d'un point de vue pratique au moins, assez facile à justifier pour les exemples relevant de la section 4.5. En effet, comme nous l'avons dit plus haut, la souplesse, la dynamique des nombres virgule flottante nous place dans un contexte où un polynôme quantifié optimal va être la plupart du temps extrêmement proche du polynôme minimax. Ici, nous nous servons alors de LLL et de l'algorithme de Babai (algorithme 4) comme d'un outil de calcul de polynômes interpolateurs à coefficients entiers.

Le cadre de la virgule fixe qui prévaut dans la section 4.6 est beaucoup plus rigide, et les fonctions de transfert quantifiées optimales sont souvent peu semblables à la fonction de transfert minimax. De plus, on travaille non plus sur un seul intervalle mais sur plusieurs, ce qui complique singulièrement la situation. Cela explique que nous ayons dû travailler plus pour élaborer au final trois stratégies de discrétisation. Nous avons ainsi mieux compris les ressorts des bons choix à faire et un élément, en particulier, a été mis en évidence : la constante de Lebesgue, cf. chapitre 3, de ces familles de points joue un rôle clé. Pour que ces discrétisations mènent à un CVP_∞ fidèle au problème initial, il est souhaitable qu'elles soient associées à une constante de Lebesgue la plus faible possible. L'algorithme 10 nous donne un procédé pour construire de telles familles et, si l'on revient au cas de l'approximation sur un seul intervalle, la famille des nœuds de Tchebychev a pour caractéristique d'être associée à une constante de Lebesgue minimale, comme indiqué par le théorème 3.36. On peut consulter [67, 66] pour voir notre étude sur l'impact de la constante de Lebesgue sur les problèmes que nous considérons.

Remplacement du CVP_∞ par un CVP_2, résolution approchée du CVP_2

Le souci de rendre nos résultats utilisables en pratique nous a amené très vite à chercher des solutions qui pourraient tirer partie de LLL, avec le bémol que ces solutions au CVP_2 donné dans (4.4) seraient a priori seulement approchées. Une conséquence du fait de privilégier cette solution approchée au CVP_2 est de diminuer l'impact du remplacement du CVP_∞ énoncé dans (4.3) par un CVP_2 : en pratique, nous utilisons notre outil de résolution approchée du CVP_2 pour nous placer dans une zone favorable, que nous allons explorer avec l'algorithme 8 pour fournir de très bons approximants quantifiés.

Enfin, pour ce qui concerne la résolution approchée du CVP_2 énoncé dans (4.4), le succès de notre approche semble venir de la nature extrêmement favorable des réseaux euclidiens que nous considérons. Comme on peut le voir dans [32, §2.3.7] et [66, §4.4.1], les bases en sortie de LLL ont des propriétés peu courantes. En effet, les vecteurs de la base de Gram-Schmidt associés à la base réduite ont des normes décroissant très lentement [66, §4.4.1] dans le cadre de la synthèse de filtres et on observe même une

croissance [32, §2.3.7] dans le cadre de la synthèse de fonction. Cela nous indique, cf. chapitre 2, que la base en sortie est formée de vecteurs presque aussi courts que possible. Dans le cadre de la synthèse de filtres, le fait que l'algorithme HKZ termine si vite sur des réseaux de dimension 62 nous indique que les réseaux de la section 4.6 sont sans doute déjà partiellement réduits dès le départ. Je ne rapporte ici que des observations expérimentales : pour le moment, je ne suis pas en mesure d'expliquer ces phénomènes si favorables.

4.7 Approximation rationnelle efficace en virgule fixe : la E-méthode

La E-méthode [61, 62], due à M. Ercegovac (Univ. of California at Los Angeles), est un processus efficace pour l'évaluation en virgule fixe de polynômes et de certains types de fractions rationnelles. Par conséquent, elle est susceptible de constituer un procédé général extrêmement intéressant pour réaliser des implantations matérielles virgule fixe très efficaces. Malheureusement, pendant longtemps, il n'existait pas d'approche systématique pour, étant donné une fonction f à évaluer sur un intervalle $[a, b]$, obtenir de bonnes approximations de f par des fractions rationnelles satisfaisant aux conditions imposées par la E-méthode.

Il y a quelques années, j'avais mis en évidence avec J.-M. Muller **C-2** des transformations qui permettent l'utilisation de la E-méthode pour la plupart des fractions rationnelles à coefficients réels.

Le problème est que les fractions rationnelles utilisables en machine ont des coefficients contraints (valeurs ou taille imposées), qui empêchent l'utilisation directe de ces transformations. S. Chevillard, M. Ercegovac, J.-M Muller, S. Torres et moi-même avons donc proposé dans **C-8** la première approche systématique permettant de calculer de bonnes approximations rationnelles d'une fonction, telles que les coefficients des fractions respectent :

— les contraintes de la E-méthode (et sont donc évaluables par cette méthode) ;

— les contraintes sur les valeurs ou la taille en bits imposées par l'application visée.

Cette nouvelle approche utilise la programmation linéaire et la réduction de réseaux.

4.8 Perspectives

Comme notre lectrice a pu le noter, l'absence d'outils efficaces pour attaquer directement CVP_∞ nous a amené à nous placer dans le cadre euclidien et considérer alors la question sous l'angle d'un CVP_2. Toutefois, l'article [43] propose un algorithme d'attaque directe du CVP_∞ que nous avons à résoudre. Je voudrais donc essayer de le mettre en œuvre et tester ses possibilités pratiques.

Je souhaite aussi continuer à creuser ces questions de qualité numérique en traitement du signal. Nous proposons actuellement une solution satisfaisante pour le cas des filtres RIF, allant jusqu'à la synthèse matérielle en FPGA de ces filtres. Il nous reste maintenant à avancer significativement sur le cas des filtres RII [6, 168, 176], dont la fonction de transfert est une fraction rationnelle, et plus un polynôme. La première difficulté est le développement de l'analogue de l'algorithme de Parks-McClellan. En effet, le calcul d'une fraction rationnelle de meilleure approximation est notoirement instable et il va nous falloir élaborer des méthodes robustes à cet effet. Peut-être aurons-nous à passer via des approximations légèrement sous-optimales mais qui nous permettront des calculs sûrs. La question de la quantification des coefficients nous posera de nouveaux problèmes puisqu'il faudra prendre en compte des contraintes additionnelles telles que la localisation des pôles afin d'assurer la stabilité du filtre.

S. Filip et moi-même projetons de concevoir une bibliothèque C++ dont l'objectif sera d'être l'outil de référence pour la synthèse de filtres RIF et RII.

Chapitre 5

Fonctions élémentaires et arrondi correct - Dilemme du fabricant de tables

Comme nous l'avons vu dans le chapitre précédent, lorsqu'on évalue une fonction telle que la racine cubique ou l'exponentielle, on évalue en fait une très bonne approximation de la fonction (un polynôme en général). Un problème se pose alors, celui de l'arrondi correct : comment garantir que l'arrondi de la valeur de la fonction va coïncider avec l'arrondi de la valeur de l'approximation ?

L'arrondi correct du résultat de l'évaluation d'une fonction élémentaire n'a pas été exigé par la version initiale du standard IEEE 754 : il a été considéré lors de la rédaction de la norme que l'arrondi correct était impossible à assurer à un coût raisonnable. Depuis plus d'une quinzaine d'années maintenant, l'un des objectifs de l'équipe Arénaire, puis de sa suite AriC, a été de montrer que tel n'est pas le cas. C'est l'objet de nombreux travaux de l'équipe et notamment de la bibliothèque `CRlibm`[1], qui permet l'évaluation avec arrondi correct des fonctions élémentaires en arithmétique double précision. Ces travaux ont contribué à ce que la dernière version de la norme recommande à présent l'arrondi correct pour les fonctions élémentaires. C'est un progrès incontestable mais il reste encore des obstacles importants avant que l'on puisse un jour passer de la recommandation à l'obligation.

Le problème principal rencontré est celui du dilemme du fabricant de tables. Lors de l'évaluation d'une fonction élémentaire, on doit effectuer les calculs avec une précision interne supérieure à la précision requise pour le résultat afin de savoir décider l'arrondi. La question qui se pose alors est de savoir quelle est la précision du calcul intermédiaire suffisante pour que l'arrondi de l'approximation coïncide toujours avec l'arrondi du résultat exact. Évidemment, on souhaite que cette précision soit minimale, afin de conserver des calculs aussi rapides que possible.

Conjointement avec G. Hanrot et O. Robert (Institut C. Jordan, Univ. St-Étienne), nous avons élaboré une approche, fondée sur les techniques de sommes d'exponentielles (un outil extrêmement fécond en théorie analytique des nombres) qui permet d'évaluer les chances d'avoir une valeur de fonction difficile à arrondir. Les estimations que nous fournissons permettent de donner un fondement solide à une heuristique utilisée depuis une trentaine d'années pour la synthèse de fonctions avec arrondi correct. Cela a donné lieu à l'article **J-14**.

Dans **J-7**, un travail commun avec J.-M. Muller, nous avons explicité le lien entre ce problème dans le cas des fonctions algébriques (telles que les racines n-èmes) et certains résultats d'approximation diophantienne pour les nombres algébriques. Cela nous permet d'obtenir des bornes, de bonne qualité, sur la précision intermédiaire des calculs nécessaire. Ce type de travaux aboutit à des optimisations importantes (parfois considérables) et permet au final d'offrir l'arrondi correct avec un surcoût acceptable pour certaines fonctions algébriques.

En collaboration avec G. Hanrot, nous avons élaboré une nouvelle approche pour le dilemme du fabricant de tables dans le cas des fonctions transcendantes, telles que l'exponentielle par exemple. Nos résultats préliminaires sont les premiers à fournir des réponses non-triviales pour le format binary128 (la précision quadruple IEEE) et ouvrent ainsi la voie à la réalisation d'une bibliothèque de fonctions élé-

1. https://gforge.inria.fr/scm/browser.php?group_id=5929&extra=crlibm

mentaires correctement arrondies dans les quatre précisions IEEE de base. Nous rédigeons actuellement l'article **S-7**. Ce travail n'étant pas encore parfaitement abouti, je l'évoquerai dans la section 5.5 sous la forme de perspectives.

Ces travaux ont été réalisés avec mes collègues Guillaume Hanrot, Jean-Michel Muller et Olivier Robert.

Les questions de l'arrondi correct et du dilemme de fabricant de tables ont été centrales dans la thèse de Serge Torres [203].

5.1 Introduction

Sur la plupart des systèmes de calcul, les nombres réels sont approchés par des nombres flottants [156]. Pendant de nombreuses années, l'arithmétique virgule flottante était régie par ce que l'on peut qualifier de recettes de cuisine, une situation décrite dans [100] par exemple. Cela a eu parfois des conséquences désastreuses et les programmes numériques n'étaient ni fiables ni portables

Le standard IEEE 754 [37, 3] qui traite de l'arithmétique virgule flottante en base 2 (ainsi que le standard IEEE 854 [36, 4], indépendant de la base choisie, qui a suivi) et sa révision [95, 144] ont amélioré la situation de manière drastique et mis un terme à cette période de grande « insécurité numérique ». En particulier, le standard IEEE 754 spécifie clairement les formats de représentation des nombres en virgule flottante, et le comportement des quatre opérations arithmétiques de base $\{+, -, \times, /\}$ et de la racine carrée.

Cependant, à ce jour, le standard n'édicte rien d'aussi précis au sujet du comportement de fonctions usuelles telles que celles de la `libm`, la bibliothèque mathématique du langage C. Les travaux que je présente dans ce chapitre font partie d'un effort d'ensemble [156] pour améliorer cette situation. Il s'agit ici de permettre l'évaluation des fonctions avec une qualité de calcul supérieure, appelée arrondi correct, que nous allons définir dès maintenant.

5.1.1 Cadre arithmétique - arrondi correct

Nous allons travailler avec les notions définies dans le chapitre 1, modulo la nuance suivante : nous allons considérer $\mathcal{D}_p$ l'ensemble des nombres flottants en base 2 avec une mantisse sur p bits, $p \geqslant 1$, et un exposant non borné :

$$\mathcal{D}_p = \left\{ j \cdot 2^{E-p+1}, 2^{p-1} \leqslant |j| \leqslant 2^p - 1, j, E \in \mathbb{Z} \right\} \cup \{0\}.$$

Cet ensemble n'est pas exactement l'ensemble des nombres flottants de la norme IEEE 754. C'est un système « idéal » de nombres flottants : c'est un sur-ensemble d'un système réel, sans débordement vers l'infini (overflow), vers zéro (underflow) ni dénormalisé (subnormals) [95, 156]. Nous prouverons nos résultats dans $\mathcal{D}_p$. Ils restent valides dans un système réel, pourvu qu'aucun débordement vers l'infini ou zéro ou une valeur dénormalisée ne soit rencontré, ce qui ici est une restriction à la fois commode et raisonnable.

La norme IEEE 754 exige que l'utilisateur ait la possibilité de choisir un des quatre modes d'arrondi, appelé alors *mode d'arrondi actif*. Une fois ce mode d'arrondi actif choisi, lorsque l'on effectue une des quatre opérations arithmétiques ou lorsqu'on calcule une racine carrée, le résultat retourné doit être égal à l'arrondi de la valeur mathématique exacte : cette exigence sur la qualité du calcul est appelée *arrondi correct*.

Si les standards IEEE 754-1985 et 854-1987 exigeaient l'arrondi correct pour ces cinq opérations, il n'en était rien pour les fonctions mathématiques les plus communes telles que les fonctions algébriques [2] $1/\sqrt{\cdot}$, $\sqrt[3]{\cdot}, \ldots$ ou des fonctions transcendantes [3] telles que le sinus, le cosinus, l'exponentielle et leurs réciproques, etc. Plus généralement, un objectif naturel est la classe des fonctions élémentaires [4]. Une

2. On dit qu'une fonction φ est algébrique s'il existe $P \in \mathbb{Z}[x, y] \setminus \{0\}$ tel que $P(x, \varphi(x)) = 0$.
3. Une fonction est dite transcendante si elle n'est pas algébrique.
4. Une fonction élémentaire est une fonction d'une variable et qui est la composée d'un nombre fini d'opérations arithmétiques $+, -, \times, /$), d'exponentielles, de logarithmes, de constantes et de fonctions algébriques.

partie de ces fonctions est usuellement disponible dans les bibliothèques mathématiques `libm` fournies avec les compilateurs.

Être capable de fournir des fonctions correctement arrondies est un enjeu important :

— cela améliore grandement la portabilité des logiciels ;
— cela permet de concevoir des algorithmes utilisant cette exigence de qualité ;
— cela permet de faire des preuves formelles de codes ;
— on peut implémenter facilement l'arithmétique d'intervalle, ou plus généralement, on peut obtenir des minorants ou des majorants garantis de la valeur exacte d'une suite d'opérations.

L'impossibilité, à l'époque de la rédaction du premier standard, d'exiger l'arrondi correct pour ces fonctions est principalement due à un problème mathématique difficile connu sous le nom du dilemme du fabricant de tables (TMD en anglais), un terme introduit par W. Kahan [101]. Pour l'illustrer, choisissons de nous placer dans le cadre de l'arrondi au plus près. On souhaite évaluer une fonction φ en un nombre flottant x. Comme dit plus haut, en général, on évalue plutôt une approximation $\widetilde{\varphi(x)}$ de $\varphi(x)$. La seule valeur dont nous disposons en pratique est $\widetilde{\varphi(x)}$ et l'on sait juste que $\varphi(x)$ se trouve à une distance au plus ε de $\widetilde{\varphi(x)}$. On veut garantir que l'arrondi au plus près $\mathrm{RN}\left(\widetilde{\varphi(x)}\right)$ coïncide avec l'arrondi au plus près $\mathrm{RN}(\varphi(x))$: cette question devient un dilemme (d'où ce terme de dilemme du fabricant de tables) si un milieu de deux nombres flottants consécutifs, autrement dit un point frontière pour l'arrondi au plus près, appartient à l'intervalle $]\widetilde{\varphi(x)} - \varepsilon, \widetilde{\varphi(x)} + \varepsilon[$. Comment lever cette ambiguïté ?

Un (bon) réflexe est de tenter de diminuer ε et de calculer une nouvelle approximation $\widetilde{\varphi(x)}_2$, distante d'au plus $\varepsilon_2 (< \varepsilon)$ de $\varphi(x)$. Il faut toutefois compléter immédiatement cette suggestion par l'observation suivante : si $\varphi(x)$ est le milieu de deux nombres flottants consécutifs, ce raffinement de l'erreur ε n'apportera aucune amélioration. On adopte donc l'approche suivante :

— on détermine au préalable si $\varphi(x)$ est le milieu de deux nombres flottants consécutifs (ou un flottant si on travaille avec un arrondi dirigé). Si c'est le cas, on renvoie son arrondi.
— sinon on diminue l'erreur d'approximation itérativement et on sait que, nécessairement, il y aura une erreur ε_n et une approximation $\widetilde{\varphi(x)}_n$ de $\varphi(x)$ telle que $]\widetilde{\varphi(x)}_n - \varepsilon_n, \widetilde{\varphi(x)}_n + \varepsilon_n[$ ne contienne aucun milieu de deux nombres flottants consécutifs (ou aucun flottant si on travaille avec un arrondi dirigé).

Cette façon de procéder est appelée *stratégie de Ziv*.

5.1.2 Implémentation de fonctions correctement arrondies : la stratégie de Ziv

A. Ziv a proposé une méthodologie générale [221], implémentée dans la bibliothèque `libultim`[5], qui a rendu possible l'évaluation avec arrondi correct des fonctions de la `libm` du langage C. Elle consiste, comme on vient de le voir, à augmenter itérativement la précision de l'approximation jusqu'à ce que l'on puisse décider de l'arrondi correct.

Cependant, il n'y avait jusqu'à il y a quelques années aucune borne pratique pour la fin des itérations de Ziv : on sait qu'elles terminent pour la plupart des fonctions transcendantes de la `libm` mais la précision maximale requise en réalité dans le pire cas était inconnue. Notons que pour prouver la terminaison de l'itération de Ziv pour une fonction réelle φ et un nombre flottant donné x, on doit garantir a priori que $\varphi(x) \notin \mathcal{D}_p$ (en mode d'arrondi dirigé) ou $\varphi(x)$ n'est pas le milieu de deux éléments consécutifs de $\mathcal{D}_p$ (pour l'arrondi au plus près). Il faut effectivement traiter séparément les cas exacts tels que $\exp(0), \log(1), \sin(0)$ ou $\sqrt[3]{x^3}$ pour $x \in \mathcal{D}_{\lfloor p/3 \rfloor}$, etc. [123, 124, 155].

Malheureusement, cela n'est pas satisfaisant pour de nombreuses applications :

— dans `libultim`, le temps d'exécution en pire cas mesuré est de trois ordres de grandeur au delà de celui des `libm`s habituelles ;
— un problème corrélé est celui de la mémoire, qui est pour les mêmes raisons non bornée théoriquement et, en pratique, bien plus importante que ce que requièrent les `libm`s ;

5. `libultim` a été publiée par IBM. Une version mise à jour fait actuellement partie de la `glibc` GNU et est disponible sous la GNU General Public License.

— quand les approximations sont évaluées à l'aide d'un multiplieur pipeliné rapide ou d'un multiplieur-accumulateur, les tests requis par la stratégie de Ziv peuvent occuper de nombreux cycles avant de redémarrer les calculs ;
— pour les applications temps-réel, le retard de calcul doit être borné.

5.1.3 Évaluation correctement arrondie rapide et peu coûteuse dans le format binary64

Quand on évalue la plupart des fonctions élémentaires, on calcule une approximation du résultat exact, avec une précision supérieure à la précision cible p. Le dilemme du fabricant de tables pose la question de la détermination de la précision nécessaire pour garantir que l'arrondi de cette approximation sera toujours l'arrondi du résultat exact. De plus, afin de pouvoir borner uniformément le nombre d'itérations de la stratégie de Ziv (telle qu'énoncée plus haut, il y a un travail spécifique à faire pour chaque flottant x, ce que nous ne souhaitons pas), nous désirons que cette précision garantissant l'arrondi correct soit uniforme : au lieu de déterminer une précision m_x dépendant de chaque flottant x, nous voulons obtenir une précision $\mu(p)$, aussi petite que possible, qui va garantir l'arrondi correct pour tous les flottants.

Ainsi, dans la suite de ce chapitre, c'est la recherche de cette précision $\mu(p)$ uniforme que nous appellerons dilemme du fabricant de tables (TMD).

Cette précision $\mu(p)$ existe bien dans $\mathbb{N}$ du fait de la finitude des nombres flottants mais a t-on une idée de sa valeur ou d'un majorant de sa valeur ?

D'un côté, des résultats théoriques donnent des informations pour les fonctions algébriques : comme on le verra dans la section 5.4, la précision à laquelle les calculs doivent être faits est en général surestimée. D'autre part, en ce qui concerne les fonctions transcendantes, soit on ne dispose d'aucun énoncé théorique soit ils nous donnent des résultats inutilisables en pratique [158].

On a donc dû développer des approches algorithmiques pour tenter de résoudre le TMD [125, 126, 197]. Elles ont permis de le faire de façon satisfaisante pour le format IEEE binary64 (aussi connu sous le nom de précision double). Une conséquence notable est que la révision du standard IEEE 754, publiée en 2008, recommande à présent (elle ne peut toutefois l'exiger, du fait de l'absence de résultats pour le format binary128) que les fonctions suivantes soient évaluées avec arrondi correct : e^x, $e^x - 1$, 2^x, $2^x - 1$, 10^x, $10^x - 1$, $\ln(x)$, $\log_2(x)$, $\log_{10}(x)$, $\ln(1+x)$, $\log_2(1+x)$, $\log_{10}(1+x)$, $\sqrt{x^2+y^2}$, $1/\sqrt{x}$, $(1+x)^n$, x^n, $x^{1/n}$ (n est un entier), $\sin(\pi x)$, $\cos(\pi x)$, $\arctan(x)/\pi$, $\arctan(y/x)/\pi$, $\sin(x)$, $\cos(x)$, $\tan(x)$, $\arcsin(x)$, $\arccos(x)$, $\arctan(x)$, $\arctan(y/x)$, $\sinh(x)$, $\cosh(x)$, $\tanh(x)$, $\sinh^{-1}(x)$, $\cosh^{-1}(x)$, $\tanh^{-1}(x)$.

Grâce à ces résultats, il est maintenant possible d'obtenir une évaluation avec arrondi correct dans le format binary64 en deux étapes de Ziv seulement, que l'on peut alors optimiser séparément. C'est l'approche utilisée dans `CRlibm`, une bibliothèque qui permet l'évaluation correctement arrondie en précision des fonctions élémentaires du standard C99 :

— la première phase, dite rapide, est aussi rapide qu'une évaluation sur une `libm` actuelle, et donne une précision de 2^{-52-k} ($k = 11$ pour la fonction exponentielle par exemple), qui est suffisante pour obtenir l'arrondi correct sur les 53 bits du format binary64 dans la plupart des cas ;
— la seconde phase, dite précise, est dédiée aux cas plus délicats. Elle est plus lente mais elle a un temps d'exécution en pire cas raisonnable, grâce à la détermination des pires cas pour l'arrondi par Lefèvre et al. [127, 126, 196, 197, 195]. En particulier, il n'y a plus besoin de précision multiple.

Cette approche [46, 48] permet de concevoir des routines d'évaluation avec arrondi correct de fonction qui sont rapides et consomment une quantité de mémoire raisonnable.

5.1.4 Une approche probabiliste heuristique

Cette stratégie s'appuie sur l'heuristique suivante, relative à la première étape : la proportion de *k-mauvais cas* (ou *mauvais cas* quand il n'y a pas d'ambiguïté), à savoir les nombres flottants dont l'évaluation ne peut pas être correctement arrondie si l'on utilise 2^{-52-k} comme précision intermédiaire (nous sommes toujours dans le format binary64), est autour de 2^{1-k}. Cette estimation a été utilisée

dans [57] et une étude probabiliste a été menée dans [70]. Nous rappelons maintenant brièvement l'intuition qui sous-tend cette heuristique (voir [155, 156] pour une présentation plus détaillée).

Soit φ une fonction à valeurs réelles, on suppose qu'après le p-ème bit, les bits de la mantisse de $\varphi(x)$, où x est un nombre flottant, sont des suites de 0 et de 1, apparaissant de manière indépendante avec une probabilité $1/2$. La probabilité que nous ayons après le bit p

— en **arrondi au plus près**, la suite de bits

$$\underbrace{100\cdots0}_{k\text{ bits}} \quad \text{ou} \quad \underbrace{011\cdots1}_{k\text{ bits}},$$

— ou, en **arrondi dirigé**, la suite de bits

$$\underbrace{00\cdots0}_{k\text{ bits}} \quad \text{ou} \quad \underbrace{11\cdots1}_{k\text{ bits}}$$

est 2^{-k+1}. Ainsi, si nous avons N nombres flottants dans le domaine que nous considérons, le nombre de valeurs x dont la mantisse admet une suite de bits d'une des formes ci-dessus est environ $N2^{-k+1}$, sous l'hypothèse probabiliste ci-dessus.

Cette approche nous donne aussi une heuristique pour le TMD : le nombre $\lceil N2^{-k+1}\rceil$ s'annule dès que k est assez grand par rapport $\log_2(2N)$. Cela signifie en pratique que k est légèrement supérieur à $\log_2(2N)$, et autrement dit que la précision intermédiaire optimale que nous cherchons à établir pour résoudre le TMD est légèrement au-dessus $p + \log_2(N) + 1$. Si l'on considère par exemple $\exp$ sur $[1/4, 1/2[$, on a $N = 2^{p-1}$ et la précision optimale (heuristique) est donc de l'ordre de 2p.

Nous pouvons maintenant formaliser les deux problèmes que nous étudions dans ce chapitre. Comme on va le voir, ils sont liés aux deux étapes de la stratégie de Ziv. Le premier problème vise à déterminer la probabilité de déclenchement de la seconde phase précise et le second n'est autre que le dilemme du fabricant de tables considéré sur une binade donnée.

5.1.5 Interprétation (encore plus heuristique) via l'approximation diophantienne

Nous avons vu plus haut que pour résoudre le dilemme du fabricant de tables pour une fonction φ, il faut déterminer $m \in \mathbb{N}$, aussi petit que possible de préférence, tel que (on utilise un arrondi dirigé, sans perte de généralité) pour tout x,

— soit $\varphi(x)$ est un flottant,

— soit on garantit qu'il n'y a aucun flottant dans l'intervalle $]\varphi(x) - 2^m, \varphi(x) + 2^m[$.

Soit $x \in [1, 2[$, nous pouvons supposer $\varphi(x) \in [1, 2[$ (il est possible de renormaliser φ). La seconde contrainte peut s'écrire :

$$\text{pour tous } 2^{p-1} \leqslant i, j \leqslant 2^p - 1, \quad \left|\varphi\left(\frac{i}{2^{p-1}}\right) - \frac{j}{2^{p-1}}\right| \geqslant \frac{1}{2^m}. \tag{5.1}$$

Ce type d'inégalités relève très précisément de l'approximation diophantienne et nous nous proposons de rappeler des résultats classiques d'approximation rationnelle.

Nous commençons par un résultat dû à Hurwitz [93]. Il suggère que pour une fonction f qui peut prendre des valeurs irrationnelles quand on l'évalue sur l'ensemble des nombres flottants, on ne peut sans doute pas espérer que la précision solution du TMD soit inférieure à $p + \log_2 N$.

Théorème 5.1. (Hurwitz) *Pour tout $\alpha \in \mathbb{R} \setminus \mathbb{Q}$, l'inégalité*

$$\left|\alpha - \frac{u}{v}\right| < \frac{1}{\sqrt{5}v^2}$$

a une infinité de solutions u et $v \in \mathbb{Z}$, $v \neq 0$. Le facteur $\sqrt{5}$ est optimal.

Ici, nous sommes intéressés par le cas où v est une puissance de 2 (2^{p-1}par exemple dans l'inégalité (5.1)). De fait, pour $\varphi = \sqrt[3]{}$ par exemple, il existe bien des valeurs de p et des flottants x dans $[1, 2^3[$ et $y = j/2^{p-1} \in [1, 2[$ tels que

$$\left| \sqrt[3]{x} - \frac{j}{2^{p-1}} \right| < \frac{1}{\sqrt{5}2^{2(p-1)}} < \frac{1}{\sqrt{5}2^{p+\log_2 N-2}} \text{ (ici, } N = 3 \cdot 2^{p-1}).$$

Le problème revient donc à déterminer combien les valeurs de f sur $\mathcal{D}_p$ peuvent être proches de nombres flottants (modes d'arrondi dirigé) ou du milieu de deux nombres flottants consécutifs (mode d'arrondi au plus près). Tout d'abord, on peut noter que l'infinité de solutions garantie par le théorème de Hurwitz possèdent la même propriété : ce sont des réduites du développement en fraction continue de α [107]. C'est un résultat classique de Legendre [128, 86].

Théorème 5.2. (Legendre) *Soit $\alpha \in \mathbb{R}$, soit u, $v \in \mathbb{Z}$, $v > 0$, tels que*

$$\left| \alpha - \frac{u}{v} \right| < \frac{1}{2v^2}$$

alors u/v est une réduite du développement en fraction continue de α.

En d'autres mots, notre problème est équivalent à déterminer les valeurs de f, en des nombres flottants, dont une des réduites est un nombre flottant ou le milieu de deux nombres flottants consécutifs, puis examiner la qualité de l'approximation rationnelle trouvée. Un résultat de Khintchine [108, 107] nous dit qu'« en général » l'approximation d'un nombre réel par les rationnels est la plus mauvaise possible.

Théorème 5.3. (Khintchine) *Soit $\varphi : \mathbb{N} \setminus \{0\} \to]0, +\infty[$, telle que $k \mapsto k\varphi(k)$ soit décroissante. Alors, pour presque tout $\alpha \in \mathbb{R}$, l'inégalité*

$$\left| \alpha - \frac{u}{v} \right| < \frac{\varphi(v)}{v}$$

n'a qu'un nombre fini de solutions u et $v \in \mathbb{Z}$, $v > 0$, si et seulement si la série de terme général $\varphi(k)$ est convergente.

En particulier, cela implique que pour presque tout $\alpha \in \mathbb{R}$, pour tout $\varepsilon > 0$, l'inégalité

$$\left| \alpha - \frac{u}{v} \right| < \frac{1}{v^{2+\varepsilon}}$$

n'a qu'un nombre fini de solutions u et $v \in \mathbb{Z}$, $v > 0$.

Si nous comparons ce résultat au théorème 5.1, cela indique que l'exposant 2 (qui correspond, en gros, à la valeur $2p$ dans le cadre du TMD) est a priori le meilleur possible.

Cette liste de résultats nous donne donc une intuition de la valeur possible pour la solution au TMD, en nous confirmant bien la valeur espérée $\approx 2p$. Il faut toutefois préciser que :

— les résultats donnés dépendent de chaque nombre réel α, alors que nous souhaitons un résultat uniforme. La contribution de chaque α n'est pas visible dans ces résultats alors qu'il faut la prendre en compte.

— Il nous faut des résultats effectifs pour pouvoir dire des choses sérieuses. Nous disposons certes d'un algorithme pour construire les réduites du développement de chaque valeur $\varphi(x)$ mais il n'est pas question de les calculer pour toutes les valeurs de φ.

Nous verrons dans la sous-section 5.4.1 un théorème classique, dû à Liouville, qui nous permet d'utiliser en pratique un résultat d'approximation rationnelle.

Remarque 5.4. Les approches algorithmiques [127, 126, 196, 197, 195], couronnées de succès dans le cas du format binary64, renforcent aussi cette heuristique $\mu(p) \approx 2p$. Par exemple, considérons la fonction $x \mapsto 2^x$ et le nombre flottant binary64

$$x = \frac{8520761231538509}{2^{62}}.$$

On a

$$\begin{aligned} 2^{52+x} &= 4509371038706515.4999999999999999994026\cdots \text{(décimal)} \\ &= \underbrace{1\cdots}_{53\text{ bits}}.0\underbrace{1\cdots\cdots\cdots\cdots 1}_{60\text{ uns consécutifs}}0\cdots \text{(binaire)}. \end{aligned}$$

C'est le cas le plus difficile à arrondir pour la fonction $x \mapsto 2^x$ en binary64. La valeur de $\mu(p)$ est de 113 ici.

5.2 Formalisation des problèmes

Supposons que l'on souhaite arrondir correctement une fonction élémentaire φ (en fait, notre formalisation est valable pour toute fonction réelle). On considère que toutes les valeurs en entrée sont des éléments de $\mathcal{D}_p$ ayant le même exposant e_1. Une analyse différente doit être faite pour chaque valeur possible de e_1.

Notons que si x est un mauvais cas pour φ, alors c'est aussi un mauvais cas pour $-\varphi$ et $-x$ est un mauvais cas pour $t \mapsto \varphi(-t)$ et $t \mapsto -\varphi(-t)$. On peut donc supposer $x \geqslant 0$ et $\varphi(x) \geqslant 0$.

Si les valeurs de $\varphi(x)$, pour $x \in [2^{e_1}, 2^{e_1+1}[$, ne sont pas toutes incluses dans la binade $[2^{e_2}, 2^{e_2+1}[$, nous décomposons l'intervalle d'entrée en sous-intervalles de telle sorte qu'il y ait un entier relatif e_2 pour lequel toutes les valeurs $\varphi(x)$, où x se promène dans le sous-intervalle, appartiennent à $[2^{e_2}, 2^{e_2+1}[$.

Nous considérons donc le cas d'un sous-intervalle I inclus dans $[2^{e_1}, 2^{e_1+1}[$.

5.2.1 Décompte des mauvais cas

Pour l'arrondi au plus près, le problème à résoudre est le suivant :

Problème 5.5 (Arrondi au plus près). *Étant donné $k \in \mathbb{N}$, estimer le nombre d'entiers X, $2^{p-1} \leqslant X \leqslant 2^p - 1$ (avec, éventuellement, les restrictions induites par la condition $X/2^{-e_1+p-1} \in I$) tels qu'il existe $Y \in \mathbb{N}$, $2^{p-1} \leqslant Y \leqslant 2^p - 1$, satisfaisant*

$$\left|\varphi\left(\frac{X}{2^{-e_1+p-1}}\right) - \frac{2Y+1}{2^{-e_2+p}}\right| < 2^{e_2-p-k}.$$

Pour les arrondis dirigés, le problème à résoudre est le suivant :

Problème 5.6 (Arrondis dirigés). *Étant donné $k \in \mathbb{N}$, estimer le nombre d'entiers X, $2^{p-1} \leqslant X \leqslant 2^p - 1$ (avec, éventuellement, les restrictions induites par la condition $X/2^{-e_1+p-1} \in I$) tels qu'il existe $Y \in \mathbb{N}$, $2^{p-1} \leqslant Y \leqslant 2^p - 1$, satisfaisant*

$$\left|\varphi\left(\frac{X}{2^{-e_1+p-1}}\right) - \frac{Y}{2^{-e_2+p-1}}\right| < 2^{e_2-p-k}.$$

5.2.2 Dilemme du fabricant de tables

Pour l'arrondi au plus près, le problème à résoudre est le suivant : *trouver $\mu(p)$, la plus petite précision pour laquelle, pour tout $x \in \mathcal{D}_p \bigcap I$ tel que $\varphi(x)$ ne soit pas le milieu de deux éléments consécutifs de $\mathcal{D}_p$ et pour tout $y \in \mathcal{D}_p$, la distance entre $\varphi(x)$ et $y + \frac{1}{2^{p-e_2}}$ soit plus grande que $2^{-\mu(p)}$.* Si l'on introduit les entiers $X = 2^{p-1-e_1}x$ et $Y = 2^{p-1-e_2}y$, on obtient

Problème 5.7 (Arrondi au plus près). *Déterminer l'entier $\mu(p) \in \mathbb{Z}$ minimum tel que, pour $2^{p-1} \leqslant X \leqslant 2^p - 1$ (avec éventuellement les restrictions induites par $x \in I$) tel que $\varphi\left(X 2^{e_1-p+1}\right)$ ne soit pas le milieu de deux éléments de $\mathcal{D}_p$ consécutifs et pour $2^{p-1} \leqslant Y \leqslant 2^p - 1$,*

$$\left|\varphi\left(\frac{X}{2^{-e_1+p-1}}\right) - \frac{2Y+1}{2^{-e_2+p}}\right| > 2^{-\mu(p)}.$$

Pour les arrondis dirigés, le problème à résoudre est le suivant : *trouver $\mu(p)$, la plus petite précision telle que, pour tout $x \in \mathcal{D}_p \bigcap I$ avec $\varphi(x) \notin \mathcal{D}_p$ et pour tout $y \in \mathcal{D}_p$, la distance entre $\varphi(x)$ et y soit plus grande que $2^{-\mu(p)}$.* Si l'on introduit les entiers $X = 2^{p-1-e_1}x$ et $Y = 2^{p-1-e_2}y$, on obtient

Problème 5.8 (Arrondis dirigés). *Déterminer l'entier $\mu(p) \in \mathbb{Z}$ minimum tel que, pour $2^{p-1} \leqslant X \leqslant 2^p - 1$ (avec éventuellement les restrictions induites par $x \in I$) tel que $\varphi\left(X2^{e_1-p+1}\right) \notin \mathcal{D}_p$ et pour $2^{p-1} \leqslant Y \leqslant 2^p - 1$,*

$$\left|\varphi\left(\frac{X}{2^{-e_1+p-1}}\right) - \frac{Y}{2^{-e_2+p-1}}\right| > 2^{-\mu(p)}.$$

5.3 Sommes d'exponentielles et proportion de mauvais cas pour l'arrondi correct

L'hypothèse sur laquelle l'approche heuristique de la section 5.1.4 repose est très forte, et, en fait, fausse en toute généralité (le cas de la fonction exponentielle autour de 0 la contredit, par exemple). Nos diverses expérimentations montrent qu'elle semble être toutefois satisfaite dans la plupart des cas. Malheureusement, la démontrer dans ces cas-là semble à ce jour hors de portée.

L'objectif de l'article **J-14** a été de donner une base théorique solide à cette heuristique pour certaines valeurs de k. Nous avons visé en particulier les valeurs de k que la bibliothèque `CRlibm` utilise en pratique. Pour donner une idée des résultats que nous pouvons atteindre, nous pouvons prouver que :

- dans le format binary64, pour $k = 11$, le nombre de mauvais cas pour
  - $\sqrt[3]{\cdot}$ sur $[1/2, 1[$ est compris entre $(1-0.058)\cdot 2^{42}$ et $(1+0.058)\cdot 2^{42}$, ce qui donne une probabilité comprise entre $(1-0.058)\cdot 2^{-10}$ et $(1+0.058)\cdot 2^{-10}$ pour le déclenchement de la seconde phase précise ;
  - $\exp$ sur $[1, 2[$ est compris entre $(1-0.054)\cdot 2^{42}$ et $(1+0.054)\cdot 2^{42}$, ce qui donne une probabilité comprise entre $(1-0.054)\cdot 2^{-10}$ et $(1+0.054)\cdot 2^{-10}$ pour le déclenchement de la seconde phase précise ;
- dans le format binary128, pour $k = 32$, le nombre de mauvais cas pour
  - $\sqrt[3]{\cdot}$ sur $[1/2, 1[$ est compris entre $(1-0.112)\cdot 2^{81}$ et $(1+0.112)\cdot 2^{81}$, ce qui donne une probabilité comprise entre $(1-0.112)\cdot 2^{-31}$ et $(1+0.112)\cdot 2^{-31}$ pour le déclenchement de la seconde phase précise ;
  - $\exp$ sur $[1, 2[$ est compris entre $(1-0.11)\cdot 2^{81}$ et $(1+0.11)\cdot 2^{81}$, ce qui donne une probabilité comprise entre $(1-0.11)\cdot 2^{-31}$ and $(1+0.11)\cdot 2^{-31}$ pour le déclenchement de la seconde phase précise.

À une renormalisation près (qui associe une fonction f à notre fonction de départ φ, voir (5.9) et (5.10)), notre problème peut être vu comme une question de décompte du nombre d'entiers m d'un intervalle $[1, M]$, tels que la distance de $f(m)$ aux entiers relatifs soit inférieure à un $\delta > 0$. Ce problème, qui revient à compter le nombre de points entiers (m, n) dans le δ-voisinage de la courbe $y = f(x)$, sera traité dans 5.3.1. Le terme principal auquel on s'attend pour ce décompte de points est $2M\delta$. L'idée de départ est d'exprimer le terme d'erreur à l'aide de parties fractionnaires, de remarquer qu'elles sont périodiques et que l'on peut donc contrôler ce terme d'erreur, de manière optimale, à l'aide d'une série de Fourier tronquée introduite par Vaaler (théorème 5.9). Le problème est alors ramené à l'estimation fine de sommes d'exponentielles, que nous effectuons grâce à l'inégalité de van der Corput (théorème 5.12). Ces techniques de sommes d'exponentielles sont au cœur de cette étude. Ces objets et la théorie sous-jacente sont des outils puissants, utilisés de manière extrêmement féconde en théorie analytique des nombres [77, 94, 151, 201, 116, 182].

Il faut noter que nos résultats sont en fait valables pour n'importe quelle fonction de classe $\mathcal{C}^2$.

Notre travail peut être considéré comme complémentaire des calculs algorithmiques de pires cas pour l'arrondi correct [125, 126, 197] de deux manières :

- alors que ces travaux donnent une analyse en pire cas pour l'implémentation d'une fonction élémentaire, nos résultats permettent une analyse en cas moyen, partielle mais rigoureuse, de cette implémentation.

— Comme on le verra dans la remarque 5.14, notre approche fonctionne, en gros, pour des valeurs de k jusqu'à $p/3$ bits. Dans cette plage, les approches algorithmiques utilisées pour attaquer le TMD sont inutilisables puisqu'il y a bien trop de mauvais cas flottants à calculer, alors que notre approche va retourner, quant à elle, une estimation fine du nombre de ces mauvais cas.

5.3.1 Estimations à l'aide de techniques de sommes d'exponentielles

Plusieurs résultats profonds (voir [19, 20, 24] par exemple) en théorie analytique des nombres reposent sur des majorations non triviales du module de sommes de la forme $\sum_{a<m\leqslant a+M} \mathrm{e}(f(m))$, où $a \in \mathbb{Z}$, $M \in \mathbb{N}$, $f : [a+1, a+M] \to \mathbb{R}$ est une fonction $\mathcal{C}^k$ avec $k \geqslant 1$ et $\mathrm{e}(t) := \exp(2i\pi t)$, $t \in \mathbb{R}$. Les techniques développées se révèlent extrêmement utiles pour estimer le nombre de points entiers dans le voisinage d'une courbe, ce qui est précisément le contexte dans lequel nous nous trouvons lorsque l'on s'attaque au dilemme du fabricant de tables. Elles sont présentées dans [77, 94, 151].

Ici, nous utilisons des techniques utilisant la dérivée seconde de f, et nous supposerons que f est de classe $\mathcal{C}^2$: si l'on vise l'estimation donnée dans le théorème 5.12, les techniques du premier ordre (à la Kuzmin-Landau) ne s'appliquent pas dans notre contexte, tandis que des méthodes d'ordre supérieur à 2 fournissent (lorsqu'elles sont applicables) des majorants moins bons.

Étant donné $0 < \eta < 1/2$, nous souhaitons établir des estimations de la valeur

$$\mathcal{R}(f;\eta) := \#\big\{1 \leqslant m \leqslant M,\ \|f(m)\| < \eta\big\}, \tag{5.2}$$

où $||x||$ désigne la distance du nombre réel x à $\mathbb{Z}$: $||x|| = \min(\{x\}, 1-\{x\})$, et $\{x\} = x - \lfloor x \rfloor$ est la partie fractionnaire x.

Quand $\eta = 0$, nous posons

$$\mathcal{R}(f;0) := \#\left\{1 \leqslant m \leqslant M,\ f(m) \in \mathbb{Z}\right\}.$$

Nous suivons l'approche développée par Vaaler dans [208]. On note $\psi(t)$ la partie fractionnaire normalisée

$$\psi(t) := t - \lfloor t \rfloor - \tfrac{1}{2}, \text{ pour tout } t \in \mathbb{R}.$$

La preuve de notre résultat principal nécessite des estimations de sommes de parties fractionnaires. Nous allons d'abord, à l'aide d'un résultat de Vaaler, approcher $\psi(t)$ à l'aide d'une série de Fourier tronquée et majorer l'erreur d'approximation à l'aide d'une autre série de Fourier tronquée. La sommation de ces deux séries tronquées va faire apparaître des sommes d'exponentielles que nous estimerons alors explicitement.

On considère la fonction

$$\widehat{J}(t) := \begin{cases} 1 & \text{si } t = 0, \\ \pi t(1-|t|)\cot(\pi t) + |t| & \text{si } 0 < |t| < 1, \\ 0 & \text{si } |t| \geqslant 1. \end{cases}$$

L'approximation de ψ que nous avons en tête est :

$$\psi_H^*(t) := -\sum_{1\leqslant |h|<H} \frac{\widehat{J}\left(\frac{h}{H}\right)}{2i\pi h}\mathrm{e}(ht), \qquad t \in \mathbb{R},\ H \in \mathbb{N}.$$

On dispose de la majoration suivante :

Théorème 5.9. *[208] Avec les notations ci-dessus, on a, pour $t \in \mathbb{R}$, $H \in \mathbb{N}$,*

$$|\psi(t) - \psi_H^*(t)| \leqslant \frac{1}{2H} \sum_{|h|<H} \left(1 - \frac{|h|}{H}\right) \mathrm{e}(ht).$$

On fait maintenant apparaître les sommes d'exponentielles.

Lemme 5.10. *Soit $\delta \in \mathbb{R}_{>0}$, et $M, H \in \mathbb{N}$. Alors pour toute fonction $f : [1, M] \to \mathbb{R}$ on a*

$$|\mathcal{R}(f;\delta) - 2M\delta| \leqslant E_1(H) + E_2(H) + \mathcal{R}(f+\delta;0) \tag{5.3}$$

et

$$|\mathcal{R}(f;\delta) - 2M\delta| \leqslant E_1(H) + 2E_2(H) \tag{5.4}$$

avec

$$E_1(H) := 2\sum_{h=1}^{H-1} \frac{|\sin(2\pi\delta h)|}{\pi h} \left|\sum_{m=1}^{M} \mathrm{e}\left(hf(m)\right)\right|,$$

$$E_2(H) := \frac{1}{H}\sum_{|h|<H}\left(1-\frac{|h|}{H}\right)\left|\sum_{m=1}^{M} \mathrm{e}\left(hf(m)\right)\right|.$$

Remarque 5.11. Notre lectrice aura peut-être remarqué que le lemme 5.10 fournit deux majorants pour une seule erreur $|\mathcal{R}(f;\delta) - 2M\delta|$. La raison en est la suivante : en pratique, nous fixerons $\delta = 2^{-k}, k \in \mathbb{N}$ et il y a des fonctions comme $\exp$ ou $\cos$ par exemple, pour lesquelles nous sommes capables d'estimer précisément la quantité $\mathcal{R}(f + 2^{-k};0)$. Dans ce cas, l'inégalité (5.3) nous donne une information plus fine que (5.4). Toutefois, cette dernière a l'avantage d'être générale.

Le théorème 5.13 est présenté de manière identique pour la même raison.

Nous énonçons maintenant une version explicite de l'inégalité de van der Corput pour les sommes d'exponentielles : c'est l'équation (6.14) de [201, p. 128].

Théorème 5.12. *Soit $M \in \mathbb{N}$, $\lambda \in \mathbb{R}_{>0}$ et $C \in [1, +\infty[$. Soit $f \in \mathcal{C}^2\big([1, M], \mathbb{R}\big)$ tel que*

$$\lambda \leqslant |f''(x)| \leqslant C\lambda \text{ pour tout } x \in [1, M].$$

Alors

$$\left|\sum_{m=1}^{M} \mathrm{e}(f(m))\right| \leqslant 3CM\lambda^{1/2} + 6\lambda^{-1/2}.$$

Donnons maintenant le résultat principal : il fournit une borne explicite qui contrôle la différence entre $\mathcal{R}(f;\delta)$ et le terme principal attendu $2M\delta$. Nous l'utiliserons dans la section 5.3.2 pour légitimer l'heuristique probabiliste de la sous-section 5.1.4 : $\mathcal{R}(f;\delta)$ dénombrera le nombre de mauvais cas et le terme principal $2M\delta$ jouera le rôle de l'estimation probabiliste. Ce résultat est en fait un corollaire d'un résultat plus général, le Theorem 3 de **J-14**. J'ai privilégié l'énoncé suivant du fait du caractère directement exploitable des bornes qu'il apporte.

Théorème 5.13. *Soit $M \in \mathbb{N} \setminus \{0\}$, $\lambda \in \mathbb{R}_{>0}$ et $C \in [1, +\infty[$. Soit $f \in \mathcal{C}^2\big([1, M], \mathbb{R}\big)$ tel que*

$$\lambda \leqslant |f''(x)| \leqslant C\lambda \qquad \text{pour tout } x \in [1, M].$$

Alors, pour tout $\delta > 0$, on a

$$\begin{aligned}|\mathcal{R}(f;\delta) - 2M\delta| \leqslant{}& \alpha_0^{2/3}C^{2/3}M\lambda^{1/3} + 2\alpha_0^{2/3}C^{2/3}\max(M\lambda^{1/3}, M^{1/2}) \\ &+ 16\alpha_0^{2/3}C^{2/3}M^{1/2} + (6+2\alpha_0)\alpha_0^{1/3}C^{4/3}M\lambda^{2/3} \\ &+ \left(\frac{60}{(2\pi)^{1/2}} + 24(2\pi)^{1/2}\delta\right)\left(\frac{\delta}{\lambda}\right)^{1/2} + \mathcal{R}(f+\delta;0)\end{aligned} \tag{5.5}$$

et

$$\begin{aligned}|\mathcal{R}(f;\delta) - 2M\delta| \leqslant{}& 2\beta_0^{2/3}C^{2/3}M\lambda^{1/3} + 4\beta_0^{2/3}C^{2/3}\max(M\lambda^{1/3}, M^{1/2}) \\ &+ 32\beta_0^{2/3}C^{2/3}M^{1/2} + (12+4\beta_0)\beta_0^{1/3}C^{4/3}M\lambda^{2/3} \\ &+ \left(\frac{60}{(2\pi)^{1/2}} + 24(2\pi)^{1/2}\delta\right)\left(\frac{\delta}{\lambda}\right)^{1/2}\end{aligned} \tag{5.6}$$

pour tout $\delta > 0$.

Remarque 5.14. Quand on met en œuvre ce résultat, on a $\delta = 2^{-k}$ et $\lambda \approx 2^{-p}$. Le terme principal $2M\delta$ vaut environ $M2^{-k+1}$ et le premier terme du reste est un $O(M2^{-p/3})$. Pour que le reste reste inférieur au terme principal, il faut donc que k reste inférieur à $p/3$. Ces estimations sont grossières (elles sont faites pour guider l'intuition de la lectrice) mais nous avons mené une analyse plus poussée dans **J-14** qui confirme cette conclusion.

Il y a deux raisons à cette limitation $k \leqslant p/3$: actuellement, nous perdons au cours de nos preuves le paramètre 2^{-k} dans certains termes du résultat du lemme 5.10 ; et c'est aussi dû à l'utilisation du théorème 5.12 qui est optimal pour les fonctions $\mathcal{C}^2$ [182].

5.3.2 Une estimation du nombre de « mauvais cas » pour la première phase rapide de la stratégie de Ziv en deux étapes

Soit $p \geqslant 1$, $e_1, e_2 \in \mathbb{Z}$, $a, b \in \mathcal{D}_p$ deux nombres flottants tels que $[a, b] \subset [2^{e_1}, 2^{e_1+1}[$, $\delta > 0$ et $\varphi : [a, b] \to \mathbb{R}$ une fonction $\mathcal{C}^\ell$, $\ell \geqslant 2$, tels que $\varphi([a, b]) \subset [2^{e_2}, 2^{e_2+1}[$[6]. Soit $A = a2^{p-1-e_1}$ et $B = b2^{p-1-e_1} \in \mathbb{N}$, et finalement $M = B - A + 1$. Afin d'attaquer les problèmes 5.5 et 5.6 de la section 5.2, nous introduisons $\mathcal{I}(\varphi; p, A, B, \delta) :=$

- $$\#\left\{A \leqslant j \leqslant B,\ \left\|2^{p-1-e_2}\varphi\left(\frac{j}{2^{p-1-e_1}}\right) - \frac{1}{2}\right\| < \delta\right\} \qquad \textbf{(arrondi au plus près)}, \tag{5.7}$$
- $$\#\left\{A \leqslant j \leqslant B,\ \left\|2^{p-1-e_2}\varphi\left(\frac{j}{2^{p-1-e_1}}\right)\right\| < \delta\right\} \qquad \textbf{(arrondis dirigés)}. \tag{5.8}$$

Dorénavant, nous appelons (A, B, δ)-mauvais cas (ou simplement mauvais cas quand il n'y a pas d'ambiguïté) les éléments de ces ensembles.

Notre objectif est d'obtenir des estimations effectives de la forme $\mathcal{I}(\varphi; p, A, B, \delta) = 2M\delta + o(M\delta)$, ou, au moins, $\mathcal{I}(\varphi; p, A, B, \delta) = 2M\delta + O(M\delta)$. Bien qu'un peu moins précise, cette dernière estimation peut fournir un majorant intéressant du nombre de mauvais cas. Rappelons que si $\delta = 2^{-k}$, $k \in \mathbb{N}$, $2M\delta$ est le nombre de mauvais cas suggérés par l'approche heuristique de la sous-section 5.1.4. Par conséquent, $\mathcal{I}(\varphi; p, A, B, 2^{-k})$ est exactement le nombre de mauvais cas pour la première phase rapide de la stratégie de Ziv à deux étapes, utilisant 2^{1-p-k} comme précision relative intermédiaire. Dans ce qui suit, nous utiliserons $\delta = 2^{-k}$ pour un $k \in \mathbb{N}$ donné (c'est le nombre de bits supplémentaires dans la première étape de la stratégie de Ziv).

Remarque 5.15. Dans le cadre de la stratégie de Ziv à deux étapes, une estimation fine de $\mathcal{I}(\varphi; p, A, B, \delta)$ permet de fournir une analyse en moyenne du coût d'évaluation de $\varphi(x)$ pour $x \in [a, b]$.

De fait, si la première étape utilise une précision relative 2^{1-p-k} et la seconde une précision relative $2^{1-p-k'}$ (avec un k' assez grand pour permettre de traiter tous les pires cas pour l'arrondi correct), le coût moyen d'une évaluation de la fonction est $c_{\varphi,p}(k) + c_{\varphi,p}(k')\mathcal{I}(\varphi; p, A, B, 2^{-k})/M$, où $c_{\varphi,p}(j)$ représente le coût d'une évaluation de f avec une précision relative 2^{1-p-j}. Pour des choix de paramètres pour lesquels nous obtenons $\mathcal{I}(\varphi; p, A, B, 2^{-k}) \approx 2^{1-k}M$, le coût moyen vaut $\approx c_{\varphi,p}(k) + 2^{1-k}c_{\varphi,p}(k')$ ($\approx$ signifie ici que nous pouvons déterminer un intervalle fin et garanti, centré en $c_{\varphi,p}(k) + 2^{1-k}c_{\varphi,p}(k')$, contenant la valeur du coût moyen).

Afin de pouvoir mettre en lien ces questions et les résultats obtenus sur (5.2),

— en **mode d'arrondi au plus près**, nous introduisons la fonction

$$f(x) := 2^{p-1-e_2}\varphi\left(\frac{A-1+x}{2^{p-1-e_1}}\right) - \frac{1}{2}, \qquad 1 \leqslant x \leqslant M, \tag{5.9}$$

et observons que $\mathcal{R}(f; \delta) = \mathcal{I}(\varphi; p, A, B, \delta)$.

— De même, en **mode d'arrondi dirigé**, nous introduisons la fonction

$$f(x) := 2^{p-1-e_2}\varphi\left(\frac{A-1+x}{2^{p-1-e_1}}\right), \qquad 1 \leqslant x \leqslant M, \tag{5.10}$$

6. Comme nous l'avons dit au début de la section 5.2, il suffit de considérer ce cas pour traiter le cas général $[a, b] \subset \pm[2^{e_1}, 2^{e_1+1}[$ et $\varphi([a, b]) \subset \pm[2^{e_2}, 2^{e_2+1}[$.

et observons encore que $\mathcal{R}(f;\delta) = \mathcal{I}(\varphi; p, A, B, \delta)$.
Nous pouvons donc à présent appliquer le théorème 5.13 pour obtenir des estimations, à la fois utiles et garanties, de $\mathcal{I}(\varphi; p, A, B, \delta)$. Noter que, quelque soit le mode d'arrondi choisi, seules les valeurs de la dérivée seconde de f importent dans le théorème 5.13 : les estimations obtenues seront donc les mêmes.

À présent, nous utilisons nos résultats sur deux exemples : la racine cubique et l'exponentielle.

5.3.3 La fonction $\sqrt[3]{\cdot}$

Dans ce cas, nous avons $\varphi(x) = x^{1/3}$. Cette fonction n'a jamais de débordement vers l'infini (overflow) ni vers zéro (underflow). Rappelons qu'elle peut prendre de nombreuses valeurs exactes : par exemple, si $x \in \mathcal{D}_{\lfloor p/3 \rfloor}$, $y = x^3 \in \mathcal{D}_p$ admet comme racine cubique exacte un flottant, à savoir x.

Un énoncé simplifié

Nous donnons ici un énoncé, conséquence des résultats plus généraux établis dans **J-14**, qui permet de visualiser deux bornes très simples, dépendant seulement des paramètres de départ p et δ.

Corollaire 5.16. *Pour la racine cubique sur la binade $[1/2, 1[$, si $p \geqslant 11$, le nombre de mauvais cas appartient à l'intervalle $[\delta 2^p - \rho, \delta 2^p + \rho]$, avec*

$$\rho \leqslant 5.72 \cdot 2^{2p/3} + 40.2 \cdot 2^{p/2} + 19.7 \cdot 2^{p/3} + 110 \cdot \delta^{1/2} 2^{p/2}.$$

Si, de plus, $p \geqslant 24$, nous avons $\rho \leqslant 8.31 \cdot 2^{2p/3} + 110 \cdot \delta^{1/2} 2^{p/2}$.

Résultats expérimentaux

Nous traitons les cas binary64 (Table 5.1) et binary128 (Table 5.2), en comparant nos bornes avec la valeur escomptée suivant le modèle probabiliste heuristique. Nous nous limitons à la binade $[1/2, 1[$, les résultats pour les autres binades étant légèrement différents, mais très semblables. Nos expériences s'appuient sur l'inégalité (5.6) du théorème 5.13.

La seconde colonne, nommée « Prévu », donne le nombre de mauvais cas prévu suivant le modèle probabiliste heuristique de la sous-section 5.1.4, à savoir $2M\delta = M2^{-k+1}$.

Quant à la troisième colonne, elle fournit une « qualité relative », autrement dit un majorant de $|\mathcal{R}(f;\delta)/2M\delta - 1|$. Par conséquent, un rayon relatif proche de 0% signifie que nos bornes donnent un ordre de grandeur très précis. D'autre part, si un rayon relatif est autour de 100%, on obtient tout de même une information intéressante, à savoir un majorant pour le nombre de mauvais cas avec le bon ordre de grandeur. Par exemple, la table 5.1 nous dit que, dans le format binary64, si $k = 14$, le nombre de mauvais cas pour $\sqrt[3]{\cdot}$ sur $[1/2, 1[$ est compris entre $(1 - 0.46) \cdot 2^{42}$ et $(1 + 0.46) \cdot 2^{42}$, ce qui donne une probabilité comprise entre $(1 - 0.46) \cdot 2^{-10}$ et $(1 + 0.46) \cdot 2^{-10}$ pour le déclenchement de la seconde phase précise.

5.3.4 La fonction $\exp$

La fonction exponentielle déborde vers l'infini pour de grandes valeurs positives de x et vers zéro pour de grandes valeurs négatives x ; d'autre part, la seule valeur rationnelle de $\exp(x)$, pour $x \in \mathbb{Q}$, est $\exp(0) = 1$, donc pour un δ rationnel, nous ignorons le terme $\mathcal{R}(f + \delta; 0)$ dans (5.5).

Un énoncé simplifié

Ici encore, nous donnons deux bornes très simples, dépendant seulement des paramètres de départ p et δ, qui sont conséquences des résultats présentés dans cette section.

Corollaire 5.17. *Pour la fonction exponentielle sur la binade $[1, 2[$, si $p \geqslant 11$, le nombre de mauvais cas est dans l'intervalle $[\delta 2^p - \rho, \delta 2^p + \rho]$, où*

$$\rho \leqslant 5.45 \cdot 2^{2p/3} + 37.2 \cdot 2^{p/2} + 27.7 \cdot 2^{p/3} + 68.2 \cdot \delta^{1/2} 2^{p/2}.$$

Si, de plus, $p \geqslant 24$, nous avons $\rho \leqslant 7.89 \cdot 2^{2p/3} + 68.2 \cdot \delta^{1/2} 2^{p/2}$.

k	Prévu	Rayon relatif
5	281474976710656	0.1%
6	140737488355328	0.18%
7	70368744177664	0.36%
8	35184372088832	0.72%
9	17592186044416	1.5%
10	8796093022208	2.9%
11	4398046511104	5.8%
12	2199023255552	11.5%
13	1099511627776	23%
14	549755813888	46%
15	274877906944	92%

TABLE 5.1 – Nos bornes pour la fonction racine cubique dans le cas binary64, $x \in [1/2, 1[$

k	Prévu	Rayon relatif
8	4.0564819e31	6.8e-7 %
11	5.0706024e30	5.4 e-6 %
14	6.3382530e29	4.3 e-5%
17	7.9228162e28	3.5 e-4%
20	9.9035203e27	2.8 e-3%
23	1.2379400e27	0.022%
26	1.5474250e26	0.18%
29	1.9342813e25	1.5%
32	2.4178516e24	11.2%
35	3.0223145e23	90%

TABLE 5.2 – Nos bornes pour la fonction racine cubique dans le cas binary128, $x \in [1/2, 1[$

Résultats expérimentaux

Nous allons maintenant illustrer nos résultats à l'aide de tables similaires à celles de la sous-section 5.3.3.

Nous présentons les bornes que nous obtenons dans les trois binades $[1, 2[$, $[32, 64[$ et $[1/64, 1/32[$, afin d'illustrer le comportement de nos bornes pour des opérandes et des valeurs de taille bien différentes.

Nos résultats sont regroupés dans les tables 5.3 et 5.4. Nous obtenons, par exemple, en binary128, que pour $k = 29$, le nombre de mauvais cas pour $\exp$ sur $[1, 2[$ est compris entre $(1 - 0.014) \cdot 2^{84}$ et $(1 + 0.014) \cdot 2^{84}$, ce qui donne une probabilité comprise entre $(1 - 0.014) \cdot 2^{-28}$ et $(1 + 0.014) \cdot 2^{-28}$ pour le déclenchement de la seconde phase précise.

On conclut cette discussion par une binade qui est intéressante pour la bibliothèque `CRlibm` puisque celle-ci commence par effectuer une réduction d'argument pour se restreindre à l'intervalle $[0, \log(2) \cdot 2^{-12}[$; nous donnons quelques-unes des valeurs obtenues par nos bornes pour la binade $[2^{-14}, 2^{-13}[$ dans la table 5.5.

5.4 Bornes pour certaines fonctions algébriques

Nous nous concentrons sur l'arrondi correct de fonctions algébriques. Nous appelons *fonction algébrique* une fonction f pour laquelle il existe un polynôme $P \in \mathbb{Z}[X, Y] \setminus \{0\}$ tel que $P(x, f(x)) = 0$. On peut prendre comme exemples :

Division, inverse $y = a/x$, avec $P(x, y) = a - xy$.

k	Prévu	Rayon relatif [1, 2[	Rayon relatif [32, 64[	Rayon relatif [1/64, 1/32[
5	281474976710656	0.09%	1%	0.02%
6	140737488355328	0.17%	1.9%	0.02%
7	70368744177664	0.34%	3.7%	0.03%
8	35184372088832	0.67%	7.3%	0.05%
9	17592186044416	1.3%	15%	0.09%
10	48796093022208	2.7%	29%	0.2%
11	4398046511104	5.4%	58%	0.3%
12	2199023255552	11%	116%	0.6%
13	1099511627776	22%	231%	1.1%
14	549755813888	43%	462%	2.0%
15	274877906944	86%	924%	3.9%

TABLE 5.3 – Nos bornes pour la fonction exponentielle dans le cas binary64

k	Prévu	Rayon relatif [1, 2[	Rayon relatif [32, 64[	Rayon relatif [1/64, 1/32[
8	4.0564819e31	6.4e-7%	6.9e-6 %	2.8e-8
11	5.0706024e30	5.1e-6%	5.6e-5%	2.2e-7
14	6.3382530e29	4.1e-5%	4.5e-4%	1.8e-6
17	7.9228162e28	3.3e-4%	3.6e-3%	1.5e-5
20	9.9035203e27	2.7e-3%	2.9e-2%	1.2e-4
23	1.2379400e27	2.1e-2%	2.3e-1%	9.0e-4
26	1.5474250e26	1.7e-1%	1.9%	7.2e-3
29	1.9342813e25	1.4%	15%	5.8e-2
32	2.4178516e24	11%	116%	0.46%
35	3.0223145e23	86%	924%	3.7%

TABLE 5.4 – Nos bornes pour la fonction exponentielle dans le cas binary128

Racines carrées $y = \sqrt{x}$, avec $P(x, y) = x - y^2$.

Racines p-èmes $y = x^{1/p}$, avec $P(x, y) = x - y^p$.

Racine carrée inverse $y = 1/\sqrt{x}$, avec $P(x, y) = 1 - xy^2$.

Lorsque $x \in \mathcal{D}_p$ et f est une fonction algébrique, la valeur $f(x)$ est un nombre algébrique, c.-à-d. la racine d'un polynôme à coefficients entiers non nul. Si α est un nombre algébrique, le polynôme minimal de α sur $\mathbb{Z}$ est le polynôme $P \in \mathbb{Z}[X] \setminus \{0\}$, à coefficients premiers entre eux et coefficient dominant positif, tel que $P(\alpha) = 0$. Si d désigne le degré de P, on dit que α est un nombre algébrique de degré d.

Pour la division, on établit immédiatement que $\mu(p) \leqslant 2p$ pour les arrondis dirigés et $\mu(p) \leqslant 2p + 1$ pour l'arrondi au plus près.

À présent, nous allons montrer comment un résultat bien connu de Liouville nous a permis d'établir dans **J-7** des bornes, comme cela avait été aussi fait dans [96] et [120], sur la solution $\mu(p)$ aux problèmes 5.7 et 5.8 pour les fonctions puissance. Nous formulons aussi quatre problèmes diophantiens dont la résolution (même partielle) permettrait une amélioration de ces bornes.

5.4.1 Quelques heuristiques et un résultat sur $\mu(p)$

Dans la sous-section 5.1.5, nous nous sommes quittés sur un résultat de Khinchine qui nous donnait un exposant d'approximation (on dit en général mesure d'irrationalité) optimal de $2 + \varepsilon$ pour presque tout nombre réel. Un résultat majeur de Roth [183] nous dit que c'est le cas des nombres algébriques.

k	Rayon relatif $[2^{-14}, 2^{-13}[$
11	14%
12	19%
13	26%
14	38%
15	54%

TABLE 5.5 – Quelques bornes dans des cas spécifiques à `CRlibm` pour la fonction exponentielle en format binary64

Théorème 5.18. (Roth) *Soit α un nombre algébrique de degré $d \geqslant 2$. Pour tout $\varepsilon > 0$, il existe $C_{\alpha,\varepsilon} > 0$ tel que, pour tout u, $v \in \mathbb{Z}$, $v \geqslant 1$,*

$$\left|\alpha - \frac{u}{v}\right| > \frac{C_{\alpha,\varepsilon}}{v^{2+\varepsilon}}. \tag{5.11}$$

Ce résultat a été étendu par D. Ridout [180] :

Théorème 5.19. (Ridout) *Soit α un nombre algébrique de degré $d \geqslant 2$. Soient $\{u_1, \ldots, u_s\}$ et $\{v_1, \ldots, v_t\}$ deux familles de nombres premiers. Soient $\sigma, \tau, c \in \mathbb{R}$ satisfaisant $0 \leqslant \sigma \leqslant 1$, $0 \leqslant \tau \leqslant 1$, $c > 1$. On suppose que les entiers relatifs u, v sont toujours de la forme*

$$u = u^* u_1^{\kappa_1} \cdots u_s^{\kappa_s},\ v = v^* v_1^{\lambda_1} \cdots v_t^{\lambda_t}, \tag{5.12}$$

où $\kappa_1, \ldots, \kappa_s, \lambda_1, \ldots, \lambda_t \in \mathbb{N}$ et $u^*, v^* \in \mathbb{Z}$ tels que

$$|u^*| < c|u|^\sigma,\ 0 < v^* < cv^\tau. \tag{5.13}$$

Alors, pour tout $\varepsilon > 0$, il existe $C'_{\alpha,\varepsilon} > 0$ tel que, pour tout u, $v \in \mathbb{Z}$ soumis aux conditions (5.12) et (5.13), on a

$$\left|\alpha - \frac{u}{v}\right| > \frac{C'_{\alpha,\varepsilon}}{v^{\sigma+\tau+\varepsilon}}. \tag{5.14}$$

Dans le cadre du TMD, on rappelle que les dénominateurs des nombres rationnels impliqués sont des puissances de 2 et l'on rentre donc dans le champ d'application du théorème de Ridout. Celui-ci semble nous suggérer donc un exposant d'approximation $1 + \varepsilon$ et donc une précision $(1 + \varepsilon)p$ pour le TMD ! Ce n'est pas le cas en pratique, pour deux raisons :

— la contribution du nombre α à cette précision nécessaire est cachée dans la constante non explicite $C'_{\alpha,\varepsilon}$.

— C'est, en quelque sorte, un résultat asymptotique. Une précision $(1 + \varepsilon)p$ va suffire mais cela sera le cas à partir d'une certaine taille, dépendant de α, de numérateurs et dénominateurs de l'approximation. Il n'y a pas de raison pour que cette taille soit de l'ordre de celle des nombres flottants que nous utilisons.

À mon sens, ces deux énoncés nous disent toutefois une vérité importante pour le TMD : les nombres algébriques sont mal approchés par les nombres rationnels (Roth) et les nombres algébriques sont encore plus mal approchés par les rationnels dyadiques (comme c'est le cas des nombres flottants) que par des rationnels quelconques. Cela a un impact sur le TMD.

Les constantes $C_{\alpha,\varepsilon}$ et $C'_{\alpha,\varepsilon}$, qui dépendent seulement de ε et α, ne sont malheureusement pas effectivement calculables. Par conséquent, ces théorèmes ne peuvent pas être utilisés pour calculer un majorant effectif pour $\mu(p)$. À ce jour, le seul résultat théorique que nous puissions exploiter est un résultat ancien (mais fondamental : il a permis d'assurer l'existence de nombres transcendants, bien avant les travaux de G. Cantor) dû à Liouville. Avant d'énoncer ce théorème, il nous reste deux choses à mentionner :

— dans le cas de la racine carrée inverse $x \mapsto 1/\sqrt{x}$ (l'adaptation aux autres fonctions puissances et racines n-èmes n'est pas difficile), Croot, Li and Zhu [41] ont montré que la conjecture abc de Masser et Oesterlé entraîne que (5.11) est vraie avec une constante indépendante de α, ce qui entraîne que $\mu(p)$ vaut bien $p + \log_2(N)$ plus « quelques bits ». Ce résultat est seulement une indication pour nous puisque la constante n'est pas, elle non plus, effectivement calculable : nous n'avons donc aucune idée de ce que représentent les « quelques bits ».
— le théorème de Liouville a été amélioré sous une forme entièrement effective par Fel'dman [65] mais pour les tailles de paramètres qui nous préoccupent, il ne nous apporte malheureusement pas une information plus fine que celle fournie par le théorème de Liouville [136, 137, 138].

Théorème 5.20. (Liouville) *Soit α un nombre algébrique de degré $d \geqslant 2$. Il existe une constante effectivement calculable C_α telle que, pour tout u, $v \in \mathbb{Z}$, $v \geqslant 1$,*

$$\left|\alpha - \frac{u}{v}\right| > \frac{C_\alpha}{v^d}.$$

La constante C_α a la forme explicite $C_\alpha = \frac{1}{\max_{|t-\alpha|\leqslant 1/2} |P'(t)|}$ où $P \in \mathbb{Z}[X]$ est le polynôme minimal de α sur $\mathbb{Z}$.

L'exposant d est moins bon (excepté dans le cas quadratique - $d = 2$ -) que celui du théorème de Roth mais le caractère effectif de la constante nous autorise des applications pratiques. Dans la sous-section 5.4.2, nous suivons donc l'approche de Liouville afin d'obtenir des bornes effectives pour la précision intermédiaire nécessaire à l'arrondi correct.

5.4.2 Fonctions puissance

Nous considérons les fonctions $t \mapsto t^{u/v}$ avec $u \in \mathbb{Z} \setminus \{0\}$ et $v \in \mathbb{N} \setminus \{0\}$, $\mathrm{pgcd}(u, v) = 1$.

On suppose que toutes les valeurs en entrée sont des flottants de $\mathcal{D}_p$ ayant même exposant e_1. Une analyse différente doit être faite pour chaque valeur possible de e_1. Pour les fonctions puissance de la forme $x^{u/v}$ il suffit de considérer v valeurs consécutives de l'exposant puisque la relation

$$(2^v x)^{u/v} = 2^u \left(x^{u/v}\right)$$

entraîne que tous les autres cas peuvent se déduire de ceux-là.

Si les valeurs de $f(x)$, pour $x \in [2^{e_1}, 2^{e_1+1}[$, ne sont pas toutes incluses dans un intervalle de la forme $[2^{e_2}, 2^{e_2+1}[$, nous décomposons l'intervalle d'entrée de sorte que sur chaque sous-intervalle, il existe un entier relatif e_2 tel que les valeurs $f(x)$, pour tout x dans ce sous-intervalle, appartiennent à $[2^{e_2}, 2^{e_2+1}[$.

Nous considérons à présent le traitement d'un sous-intervalle I inclus dans $[2^{e_1}, 2^{e_1+1}[$.

Cas où u est positif

Pour $k \in \{0, \ldots, u-1\}$, soit $x \in [2^{kv/u}, 2^{(k+1)v/u}[\cap \mathcal{D}_p$, soit $y \in [2^k, 2^{k+1}[$. Soit $j_k \in \mathbb{N}$, $\lfloor kv/u \rfloor \leqslant j_k \leqslant \lceil (k+1)v/u \rceil - 1$ tel que $x \in [2^{j_k}, 2^{j_k+1}[$.

Nous supposons que $f(x) \notin \mathcal{D}_p$ dans le cas des arrondis dirigés et $f(x)$ n'est pas le milieu de deux éléments consécutifs de $\mathcal{D}_p$ dans le cas de l'arrondi au plus près.

Le nombre algébrique $x^{u/v}$ est racine du polynôme $Q(t) = t^v - x^u$.

Arrondis dirigés

Le théorème des accroissements finis nous donne l'existence d'un c compris strictement entre $x^{u/v}$ et y tel que

$$Q(y) - Q\left(x^{u/v}\right) = Q'(c)\left(y - x^{u/v}\right)$$

c.-à-d.

$$y^v - x^u = vc^{v-1}\left(y - x^{u/v}\right).$$

D'où

$$\begin{aligned}\left|y - x^{u/v}\right| &= \frac{1}{vc^{v-1}}\left|\frac{Y^v}{2^{v(p-1-k)}} - \frac{X^u}{2^{u(p-1-j_k)}}\right| \\ &= \frac{1}{vc^{v-1}}\left|\frac{2^{kv+\max(u-v,0)(p-1)}Y^v - 2^{uj_k+\max(v-u,0)(p-1)}X^u}{2^{\max(u,v)(p-1)}}\right| \\ &= \frac{2^{\min(a,b)}}{vc^{v-1}}\frac{\left|2^{a-\min(a,b)}Y^v - 2^{b-\min(a,b)}X^u\right|}{2^{\max(u,v)(p-1)}} \qquad (5.15)\end{aligned}$$

avec $a = kv + \max(u-v,0)(p-1)$ et $b = uj_k + \max(v-u,0)(p-1)$.

Comme $c < 2^{k+1}$, l'égalité (5.15) implique

$$\left|y - x^{u/v}\right| > \frac{2^{\min(a,b)}}{v2^{(v-1)(k+1)}}\frac{\left|2^{a-\min(a,b)}Y^v - 2^{b-\min(a,b)}X^u\right|}{2^{\max(u,v)(p-1)}}.$$

L'entier $2^{a-\min(a,b)}Y^v - 2^{b-\min(a,b)}X^u$ est non nul puisque l'on a supposé $\left|y - x^{u/v}\right| \neq 0$. Nous obtenons donc le majorant

$$\begin{aligned}\mu(p) \leqslant -\log_2\left(\left|y - x^{u/v}\right|\right) \leqslant \max(u,v)(p-1) + (v-1)(k+1) + \log_2(v) \\ - \min(kv + \max(u-v,0)(p-1), uj_k + \max(v-u,0)(p-1)).\end{aligned}$$

Cette analyse fournit aussi le problème diophantien associé suivant :

Problème 5.21 ($x^{u/v}$ pour $x \in [1, 2^v[$, arrondis dirigés). *Pour $k \in \{0, \ldots, u-1\}$, pour $j_k \in \mathbb{N}$, $\lfloor kv/u \rfloor \leqslant j_k \leqslant \lceil (k+1)v/u \rceil - 1$, trouver deux entiers $X \in [2^{j_k+p-1}, 2^{j_k+p}[$ et $Y \in [2^{k+p-1}, 2^{k+p}[$ qui minimisent*

$$\left|2^{a-\min(a,b)}Y^v - 2^{b-\min(a,b)}X^u\right|$$

avec $a = kv + \max(u-v,0)(p-1)$ et $b = uj_k + \max(v-u,0)(p-1)$.

Arrondi au plus près

On obtient

$$\mu(p) \leqslant \max(u,v)(p-1) + (v-1)(k+1) + \log_2(v) + v - \min(a,b) - \min(v, \max(a-b,0)),$$

et le problème diophantien associé est :

Problème 5.22 ($x^{u/v}$ pour $x \in [1, 2^v[$, arrondi au plus près). *Pour $k \in \{0, \ldots, u-1\}$, pour $j_k \in \mathbb{N}$, $\lfloor kv/u \rfloor \leqslant j_k \leqslant \lceil (k+1)v/u \rceil - 1$, trouver deux entiers $X \in [2^{j_k+u-1}, 2^{j_k+u}[$ et $Y \in [2^{k+u-1}, 2^{k+u}[$ qui minimisent*

$$\left|(2Y+1)^v 2^{a-\min(a,b)}/D - X^u 2^{v+b-\min(a,b)}/D\right|$$

avec $a = kv + \max(u-v,0)(p-1)$, $b = uj_k + \max(v-u,0)(p-1)$ et $D = 2^{\min(v,\max(a-b,0))}$.

Cas où u est négatif

Pour $k \in \{0, \ldots, |u|-1\}$, soit $x \in [2^{kv/|u|}, 2^{(k+1)v/|u|}[\cap \mathcal{D}_p$, soit $y \in]2^{-k-1}, 2^{-k}]$. Soit $j_k \in \mathbb{N}$, $\lfloor kv/|u| \rfloor \leqslant j_k \leqslant \lceil (k+1)v/|u| \rceil - 1$ tel que $x \in [2^{j_k}, 2^{j_k+1}[$. On suppose que $f(x) \notin \mathcal{D}_p$ dans le cas des arrondis dirigés et $f(x)$ n'est pas le milieu de deux éléments consécutifs de $\mathcal{D}_p$ dans le cas de l'arrondi au plus près.

Ici encore, nous remarquons que $x^{u/v}$ est racine du polynôme $Q(t) = t^v - x^u$.

Arrondis dirigés

On obtient le majorant suivant pour $\mu(p)$:

$$\mu(p) \leqslant vp + k + \log_2(v) + |u|(p - 1 - j_k),$$

et le problème diophantien associé est :

Problème 5.23 ($x^{u/v}$ pour $x \in [1, 2^v[$, arrondis dirigés). *Pour $k \in \{0, \ldots, |u| - 1\}$, pour $j_k \in \mathbb{N}$, $\lfloor kv/|u| \rfloor \leqslant j_k \leqslant \lceil (k+1)v/|u| \rceil - 1$, trouver deux entiers $X \in [2^{j_k+p-1}, 2^{j_k+p}[$ et $Y \in [2^{k+p}, 2^{k+p+1}[$ qui minimisent*

$$\left| X^{|u|} Y^v - 2^{v(p+k)+|u|(p-1-j_k)} \right|.$$

Arrondi au plus près

On obtient

$$\mu(p) \leqslant v(p+1) + k + \log_2(v) + |u|(p - 1 - j_k).$$

et le problème diophantien associé est

Problème 5.24 ($x^{u/v}$ pour $x \in [1, 2^v[$, arrondi au plus près). *Pour $k \in \{0, \ldots, |u| - 1\}$, pour $j_k \in \mathbb{N}$, $\lfloor kv/|u| \rfloor \leqslant j_k \leqslant \lceil (k+1)v/|u| \rceil - 1$, trouver deux entiers $X \in [2^{j_k+p-1}, 2^{j_k+p}[$ et $Y \in [2^{k+p}, 2^{k+p+1}[$ qui minimisent*

$$\left| X^{|p|} (2Y + 1)^v - 2^v r^{v(p+k)+|u|(p-1-j_k)} \right|.$$

5.5 Perspectives

En collaboration avec O. Robert, nous souhaitons approfondir l'étude de **J-14** présentée dans ce chapitre. Les résultats que nous avons obtenus restent partiels (je fais référence à la limitation $k \leqslant p/3$) et il y a un travail mathématique, a priori difficile, à faire sur certaines intégrales oscillantes [199] pour améliorer la situation.

Enfin, comme mentionné dans l'introduction, j'ai entamé avec Guillaume Hanrot le développement d'une approche algorithmique du dilemme du fabricant de tables pour les fonctions transcendantes **S-7**. Nous nous focalisons sur l'exemple central de la fonction exponentielle. Les meilleures bornes théoriques, démontrées dans [158] par Nesterenko et Waldschmidt, sont inutilisables en pratique et il faut donc procéder actuellement à une attaque algorithmique, à l'instar de l'algorithme développé par V. Lefèvre pendant sa thèse [127, 125, 126] et de l'algorithme SLZ [196, 197, 195]. L'algorithme de Lefèvre a permis de régler le cas de la précision binary64 mais sa complexité en $O(2^{p/3})$ le rend inopérant pour trouver les pires pour l'arrondi correct en binary128. L'algorithme SLZ atteint une meilleure complexité, soit $O(2^{p/2})$, lors de cette recherche de pires cas mais elle reste malheureusement prohibitive pour le format binary128.

Nous avons mis au point une approche qui consiste à construire deux polynômes à deux variables qui capturent les pires cas sur un intervalle (dont la taille est imposée par une précision cible. Par exemple, cette précision vaut $2p$ si l'on souhaite déterminer les pires cas). Un relèvement de Hensel nous permet d'obtenir ces valeurs ou de prouver qu'elles n'existent pas. Les coefficients de ces polynômes proviennent, de manière analogue à l'approche présentée dans le chapitre 4, de la résolution approchée d'un SVP$_\infty$ (au lieu d'un CVP$_\infty$ dans le chapitre 4). Quant au réseau associé, il est construit à l'aide de nœuds de Tchebychev, cf. chapitre 3, qui nous permettent d'imposer une contrainte de petitesse uniforme aux deux polynômes que l'on détermine.

Notre méthode a certains traits communs avec l'algorithme SLZ. En particulier, elle a aussi une complexité en $O(2^{p/2})$ pour le calcul des pires cas et ne nous permet donc pas d'apporter une réponse optimale au TMD, soit $\mu(p) \approx 2p$, pour le format binary128. Voilà pour les mauvaises nouvelles. La bonne nouvelle provient du fait que si l'on cherche à prouver quelque chose d'un peu plus faible - mais toujours utilisable en pratique - tel que $\mu(p) \leqslant 6p$ ou $\mu(p) \leqslant 8p$ pour $p = 113$, la précision du format binary128, notre méthode permet de le faire (contrairement à l'algorithme SLZ, comme Serge Torres l'a

montré dans sa thèse [203]). C'est un résultat nouveau et qui laisse entrevoir la possibilité de concevoir une bibliothèque mathématique, avec arrondi correct, performante pour le format binary128. C'était une tâche hors de portée jusqu'à présent.

Outre la poursuite de la mise au point de cet article **S-7**, nous souhaitons déployer à grande échelle le procédé qu'il expose.

Chapitre 6

Approximations polynomiales rigoureuses

Pour garantir l'arrondi correct ou l'arrondi fidèle (voir p.36 pour une définition), il nous faut être capable d'évaluer finement et de manière certaine la norme sup d'une erreur de la forme $f - p$ (erreur absolue) ou $1 - p/f$ (erreur relative) où f est la fonction évaluée et p l'approximant considéré. Précisons que cette question d'optimisation globale a en fait une portée bien plus générale et est une primitive-clé du calcul certifié. Le stage de M2 de Mioara Joldeş (effectué sous la direction de J.-M. Muller et moi-même en 2008. Voir l'annexe C et [98] pour la thèse de Mioara) a été l'occasion d'étudier et d'implanter dans l'outil `Sollya` des méthodes probabilistes et déterministes de calcul de normes sup de fonctions, de classe $\mathcal{C}^2$ en une et plusieurs variables. Les méthodes probabilistes permettent d'obtenir rapidement une valeur pertinente mais non certifiée de ces normes sup. Cela se révèle utile au cours de l'exécution de l'algorithme de Remez, qui calcule les meilleures approximations minimax. Les méthodes certifiées [106, 178, 148] étudiées dans le mémoire de Mioara sont à base de techniques de séparation et évaluation (branch-and-bound en anglais) et utilisent l'arithmétique d'intervalles. Elles donnent des résultats rapides et pertinents dans de nombreux cas mais malheureusement, elles se révèlent lentes et peu fines quand il s'agit de calculer les normes sup de fonctions d'erreur que nous rencontrons dans nos problèmes d'évaluation de fonction. Cela est dû à l'excellente qualité d'approximation utilisée qui aboutit à des phénomènes numériques, appelés « cancellations », qui se révèlent désastreux pour le fonctionnement des approches classiques.

S. Chevillard, M. Joldeş et C. Lauter, ont mis au point dans `Sollya` des outils de différentiation automatique aux applications très prometteuses : pour les exemples testés, ils obtiennent rapidement (une poignée de secondes au plus) une estimation fine et certifiée des normes infinies de fonctions d'erreur, et cela avec des polynômes de degré 50 par exemple, sachant que les méthodes classiques plafonnent à des degrés de l'ordre 10. Ce travail [34] a été publié dans les actes d'ARITH, la conférence de référence d'arithmétique des ordinateurs. Depuis, les mêmes auteurs, en collaboration avec J. Harrison (INTEL Portland), ont publié [33], une version améliorée et étendue de ce travail.

Pendant que je supervisais ces travaux, M. Joldeş et moi avons eu l'idée, en nous inspirant de travaux antérieurs de Makino et Berz [142], d'introduire des outils nouveaux pour la validation d'erreurs d'approximation. Il s'agit de représenter une fonction à l'aide d'un couple, que nous avons appelé « modèle de Tchebychev », formé d'un polynôme d'interpolation aux nœuds de Tchebychev et d'un intervalle contenant toutes les valeurs prises par la différence entre la fonction et son polynôme d'interpolation. L'idée est de commencer par calculer les modèles de Tchebychev de certaines fonctions de base (comme les fonctions algébriques ou élémentaires) à l'aide d'arithmétique d'intervalles et de différentiation automatique, et ensuite d'élaborer une arithmétique performante permettant d'obtenir à partir des modèles de Tchebychev des fonctions de base, un modèle de Tchebychev d'une somme, d'un produit, d'une composée de ces fonctions de base. On obtient ainsi, rapidement, une approximation très fine et validée de nombre de fonctions courantes. Ce travail a été publié dans **C-12**.

De façon plus générale, nous avons nommé approximation polynomiale rigoureuse d'une fonction

f sur un certain domaine E tout couple (P, Δ) où P est un polynôme généralisé (on considère des polynômes, des polynômes trigonométriques, des combinaisons linéaires de fonctions éléments d'une base hilbertienne, ...) et Δ un domaine reste tel que $f(x) - P(x) \in \Delta$ pour tout $x \in E$.

Avec plusieurs collègues, nous avons prouvé formellement plusieurs résultats sur ces approximations polynomiales rigoureuses. Ce travail est décrit dans **C-14**.

Dans un travail commun avec F. Bréhard et M. Joldeş, nous proposons une méthode en complexité linéaire pour le calcul d'approximations polynomiales rigoureuses de la solution d'une équation différentielle à coefficients assez réguliers (ce cas comprend celui des fonctions analytiques et a fortiori, des polynômes) avec conditions initiales. Nous sommes en train de terminer **S-5** que j'évoquerai brièvement dans les perspectives de travail.

L'essentiel des travaux évoqués dans ce chapitre ont été (ou sont) réalisés avec ma collègue Mioara Joldeş et notre étudiant Florent Bréhard. Ils sont au cœur de leurs thèses de doctorat [98].

La petite partie dévolue à la preuve formelle a été réalisée avec mes collègues M. Joldeş, É. Martin-Dorel, M. Mayero, J.-M. Muller, I. Paşca, L. Rideau et L. Théry.

6.1 Introduction

Je présente maintenant deux questions bien représentatives de ce qui a motivé notre étude.

Problème 6.1. *Imaginons qu'au cours du processus d'implémentation de la fonction $f(x) = e^{1/\cos x}, x \in [0, 1]$ en logiciel ou en matériel, nous soyons amenés à encadrer finement et de manière garantie l'erreur d'approximation $\varepsilon = f - P^*$, où P^* est l'approximation polynomiale minimax de f de degré au plus 10. Si l'on applique naïvement les techniques de base de l'arithmétique d'intervalle (cf. section 6.2) en remplaçant chaque élément composant ε par son image et en combinant ces images de manière élémentaire, on obtient $\|\varepsilon\|_\infty \leqslant 298$. Une idée usuelle dans ce domaine est de sous-diviser l'intervalle de départ en plusieurs intervalles sur lesquels l'idée naïve donnera des résultats moins grossiers. Malheureusement, pour obtenir la vraie valeur $\|\varepsilon\|_\infty = 3.83\ldots \cdot 10^{-5}$, il faudrait travailler avec 10^7 sous-intervalles. Des méthodes déterministes plus sophistiquées [106, 178, 148] sont elles aussi mises en échec.*

Passons maintenant à un autre problème, celui de la quadrature certifiée.

Problème 6.2. *L'intégration numérique est une routine commune à tous les logiciels de calcul numérique ou symbolique tels que Maple, MATLAB, Mathematica, SAGE, etc. Aucun de ces outils n'offre de garantie sur le résultat renvoyé. De fait, il y a quelques années, M. Joldeş a été contactée au cours de sa thèse par notre collègue A. Goldsztejn (CNRS, Nantes) qui lui avait soumis l'intégrale suivante :*

$$J = \int_0^3 \sin\left(\frac{1}{(10^{-3} + (1-x)^2)^{3/2}}\right) \mathrm{d}x.$$

A. Goldsztejn, intrigué par certains des résultats annoncés dans [30] qui présente des méthodes de calcul certifié d'intégrales, souhaitait voir si nos outils pouvaient l'aider à tester ces résultats de manière fiable.

Comme on le voit sur la figure 6.1, la fonction intégrée est extrêmement oscillante. Voilà les réponses fournies par des outils courants :

- *Maple18 :* 0.7499743685, *Maple2016 ne termine pas ;*
- *Pari/GP :* 0.7927730971479080755 *;*
- *Mathematica and Chebfun ne répondent pas ;*
- *SAGE :* 0.7499743685 *;*
- *Chen [30] :* 0.7578918118.

La réponse à ces deux problèmes peut être donnée par le calcul d'approximations polynomiales rigoureuses :

Définition 6.3. *Soit $f \in \mathcal{C}([a, b])$. Une approximation polynomiale rigoureuse de f est un couple (p, Δ) où $p \in \mathbb{R}[x]$ et Δ un intervalle fermé tels que $f(x) - p(x) \in \Delta$ pour tout $x \in [a, b]$.*

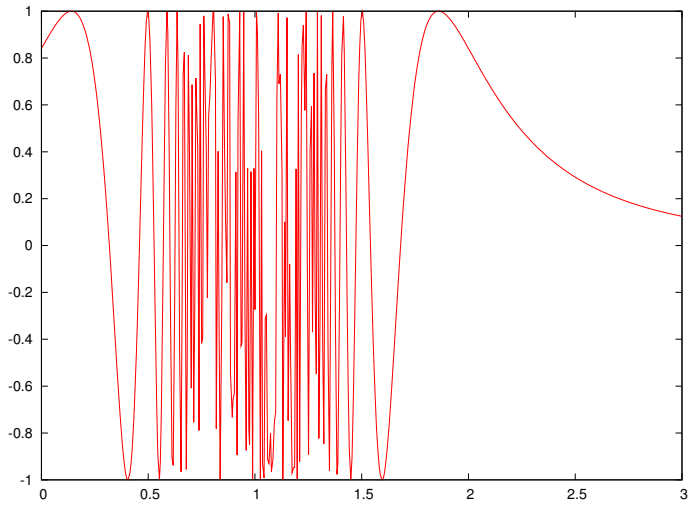


FIGURE 6.1 – La fonction $x \mapsto \sin\left(\frac{1}{(10^{-3}+(1-x)^2)^{3/2}}\right)$

Bien sûr, il faut que ce calcul soit pratique et qu'il nous fournisse un intervalle Δ fin. Le premier procédé significatif pour réaliser cela a été fourni par les travaux de Moore [152] et Makino et Berz [142, 15, 16] qui ont, respectivement, proposé l'idée des modèles de Taylor et les ont mis au point en pratique. Nous présenterons les modèles de Taylor, et ce qui nous a poussés à tenter de leur trouver une alternative, dans la section 6.3. Ensuite, nous introduirons dans la section 6.4 les modèles de Tchebychev, des approximations polynomiales rigoureuses plus robustes et précises que les modèles de Taylor qui ont été proposées par M. Joldeş et moi dans **C-12**. Nous mettrons en œuvre nos outils sur une série d'exemples, et répondrons au passage aux problèmes 6.1 et 6.2 dans la section 6.5. Enfin, nous évoquerons brièvement la formalisation en COQ de ces travaux dans la section 6.6 et conclurons ce chapitre par une série de perspectives.

Au préalable, nous présentons rapidement l'arithmétique et l'analyse d'intervalles dans la section 6.2. Elles constituent l'outil de base pour le calcul rigoureux.

6.2 Arithmétique d'intervalles, analyse d'intervalles

L'arithmétique d'intervalles est une « une arithmétique pour les inégalités ». Supposons par exemple que nous avons $5 \leqslant a \leqslant 6$ et $10 \leqslant b \leqslant 11$, cela entraîne $50 \leqslant ab \leqslant 66$. Nous allons définir, avec d'autres opérations, un produit sur les intervalles réels tel que $[5, 6] \times [10, 11] = [50, 66]$. Outre cette approche ensembliste assez naturelle, une autre motivation pour une arithmétique d'intervalles provient de la nécessité de la gestion des erreurs d'arrondi qui se produisent dès que l'on travaille en précision finie.

L'arithmétique et l'analyse d'intervalles sont des éléments-clés du calcul rigoureux en machine. Ces dernières années, des résultats de première importance ont utilisé l'ordinateur et l'arithmétique d'intervalles dans certains points de leur démonstration. On peut citer comme exemples la preuve par T. Hales de la conjecture de Kepler [82, 83] et la solution au 14-ème problème de Smale par W. Tucker [205, 206].

La lectrice intéressée trouvera de très bonnes présentations du sujet dans les ouvrages [152, 153, 159, 207] et pourra aussi consulter avec interêt le site `http://www.cs.utep.edu/interval-comp/`.

6.2.1 Arithmétique d'intervalles

On ne travaille qu'avec des intervalles fermés :

Définition 6.4. *(Intervalle réel) Soit $\underline{x}, \bar{x} \in \mathbb{R}$, $\underline{x} \leqslant \bar{x}$. On définit l'intervalle*

$$X = [\underline{x}, \bar{x}] = \{x \in \mathbb{R}, \underline{x} \leqslant x \leqslant \bar{x}\}.$$

L'ensemble de tous les intervalles réels sera noté $\mathbb{IR}$.

Opérations sur les intervalles

Définissons maintenant les opérations arithmétiques de base sur les intervalles. La lectrice remarquera que la monotonie joue un rôle essentiel.

Définition 6.5. *Soit $X, Y \in \mathbb{IR}$. Soit $* \in \{+, -, \times, /\}$. On note*

$$X * Y = \{x * y; x \in X, y \in Y\}$$

où, si $* = /$, l'on suppose que $0 \notin Y$.

Proposition 6.6. *On peut calculer $X * Y$ à l'aide des formules*

$$\begin{aligned}
[\underline{x}, \bar{x}] + [\underline{y}, \bar{y}] &= \left[\underline{x} + \underline{y}, \bar{x} + \bar{y}\right], \\
[\underline{x}, \bar{x}] - [\underline{y}, \bar{y}] &= \left[\underline{x} - \bar{y}, \bar{x} - \underline{y}\right], \\
[\underline{x}, \bar{x}] \times [\underline{y}, \bar{y}] &= \left[\min\left(\underline{x} \cdot \underline{y}, \underline{x} \cdot \bar{y}, \bar{x} \cdot \underline{y}, \bar{x} \cdot \bar{y}\right), \max\left(\underline{x} \cdot \underline{y}, \underline{x} \cdot \bar{y}, \bar{x} \cdot \underline{y}, \bar{x} \cdot \bar{y}\right)\right], \\
[\underline{x}, \bar{x}] / [\underline{y}, \bar{y}] &= [\underline{x}, \bar{x}] \times \left[\frac{1}{\bar{y}}, \frac{1}{\underline{y}}\right] \quad \text{si } 0 \notin Y,
\end{aligned}$$

qui dépendent seulement des bornes des intervalles.

Remarque 6.7. Dans $\mathbb{IR}$, les opérations $+$ et $\times$ sont associatives et commutatives.

Remarque 6.8. Il peut être commode de définir un résultat pour la division dans le cas où $0 \in Y$. On trouve une discussion intéressante sur cette question dans [207, §2.3]. Pour cela, on travaille sur $\overline{\mathbb{R}}$=$\mathbb{R} \cup \{+\infty, -\infty\}$, avec deux zéros signés $+0$ et -0 (plus précisément : sur $\mathbb{R} \backslash \{0\} \cup \{+\infty, -\infty, +0, -0\}$, avec deux zéros signés $+0$ et -0).

Maintenant, il est temps d'énoncer quelques résultats un peu désagréables (voir sous-section 6.2.3 pour la raison de cette appréciation).

Proposition 6.9.

1. *La soustraction d'intervalles n'est pas l'inverse de l'addition.*
2. *La division d'intervalles n'est pas l'inverse de la multiplication.*
3. *La multiplication d'un intervalle par lui-même ne revient pas à élever au carré l'intervalle, c.-à-d. en général,*
$$[\underline{x}, \bar{x}] \times [\underline{x}, \bar{x}] \neq \left[\min\left(\underline{x}^2, \bar{x}^2\right), \max\left(\underline{x}^2, \bar{x}^2\right)\right].$$
4. *La multiplication d'intervalles est sous-distributive par rapport à l'addition : pour tous $X, Y, Z \in \mathbb{IR}$, nous avons*
$$X \times (Y + Z) \subset X \times Y + X \times Z.$$
5. *Pour tout $X \in \mathbb{IR}$, nous avons $X + [0] = X$ et $[0] \times X = [0]$.*

Remarque 6.10. Lorsqu'on implémente en machine l'arithmétique d'intervalles, les bornes des intervalles avec lesquels on travaille sont alors des nombres flottants. Il apparaît une difficulté supplémentaire puisque l'ensemble des nombres flottants n'est pas clos pour les opérations arithmétiques de base (par exemple, la somme de deux nombres flottants n'est pas toujours un nombre flottant). On doit donc arrondir avec soin, en s'assurant que l'intervalle dont les bornes sont les arrondis contient toujours l'intervalle à bornes réelles résultant de l'opération effectuée.

Remarque 6.11. Quand on fait des calculs rigoureux, la précision flottante ambiante est souvent insuffisante (si l'on travaille avec des intervalles de bornes très petites par exemple). On a donc recours à l'arithmétique flottante multiprécision. Un exemple de bibliothèque qui permet de réaliser ce type de calculs est `MPFR`[1], sur laquelle s'appuie la bibliothèque d'arithmétique d'intervalles `MPFI`[2].

1. http://www.mpfr.org
2. https://gforge.inria.fr/frs/?group_id=157

6.2.2 Fonctions d'intervalles

Trouver l'image exacte d'une fonction est en général un problème difficile. On se contente donc souvent d'une surestimation (c.-à-d. un sur-intervalle) de cette image exacte. Il y a toutefois quelques cas favorables que nous regroupons ici.

Soit $X = [\underline{x}, \bar{x}] \in \mathbb{IR}$. Grâce à la monotonie, les fonctions d'intervalles définies ci-après donnent l'image exacte des fonctions correspondantes :

$$\begin{aligned} e^X &= [\exp \underline{x}, \exp \bar{x}]\,, \\ \sqrt{X} &= \left[\sqrt{\underline{x}}, \sqrt{\bar{x}}\right], \qquad \underline{x} \geqslant 0, \\ \log X &= [\log \underline{x}, \log \bar{x}]\,, \qquad \underline{x} > 0, \\ \arctan X &= [\arctan \underline{x}, \arctan \bar{x}]\,, \end{aligned}$$

Pour d'autres fonctions telles que x^n, les fonctions trigonométriques, etc., donner l'image exacte est bien entendu possible du moment que l'on connaît leurs extrema. Par exemple, soit $n \in \mathbb{Z}$, $X \in \mathbb{IR}$,

$$X^n = \mathrm{pow}(X, n) = \begin{cases} \text{si } n \in 2\mathbb{N}+1, [\underline{x}^n, \bar{x}^n] \\ \text{si } n \in \mathbb{N} \setminus \{0\}, n \text{ pair}, [\min(\underline{x}^n, \bar{x}^n), \max(\underline{x}^n, \bar{x}^n)] \text{ si } 0 \notin X, \\ \qquad\qquad [0, \max(\underline{x}^n, \bar{x}^n)] \text{ sinon}, \\ [1,1] \text{ si } n = 0, \\ [1/\bar{x}, 1/\underline{x}]^{-n} \text{ si } -n \in \mathbb{N} \text{ et } 0 \notin X. \end{cases}$$

On a des formules analogues pour $\sin, \cos, \tan$.

6.2.3 Le phénomène de dépendance

Reprenons le point 1 de la proposition 6.9. Si l'on a évidemment $x - x = 0$ pour tout $x \in [-2, 3]$, on a en revanche $[-2, 3] - [-2, 3] = [-5, 5]$: la substitution directe des intervalles aux variables peut donner lieu à des surestimations grossières. Ce phénomène, que l'on appelle « problème de dépendance », peut parfois s'atténuer en découpant l'intervalle initial en un certain nombre des sous-intervalles et en travaillant sur chaque sous-intervalle. Toutefois, à l'instar de ce qu'il se passe dans le problème 6.1 présenté dans l'introduction, il y a de nombreux cas [15, 16, 34, 33] où le nombre de sous-intervalles nécessaires rend l'approche prohibitive.

6.3 Modèles de Taylor

Le point de départ est très simple : il nous faut un polynôme et un reste garanti, ce que les formules de Taylor nous fournissent si la fonction cible est suffisamment régulière :

Lemme 6.12. *Soit $n \in \mathbb{N}$, et soit $f \in \mathcal{C}^{n+1}([a, b])$. Soit $x_0 \in [a, b]$. Pour tout $x \in [a, b]$, nous avons*

$$f(x) = \sum_{i=0}^{n} \frac{f^{(i)}(x_0)}{i!}(x - x_0)^i + \underbrace{\int_{x_0}^{x} \frac{f^{(n+1)}(t)}{n!}(x - t)^n \mathrm{d}t}_{R_n(x)}.$$

On appelle cet énoncé formule de Taylor avec reste intégral.

En particulier, il existe ξ_x entre x_0 et x (donc dans $]a, b[$) tel que

$$R_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}(x - x_0)^{n+1} \textit{(formule de Taylor-Lagrange)}.$$

Pour obtenir une approximation polynomiale rigoureuse à partir de ce lemme, il y a deux types de calculs à mener :

— celui des coefficients de Taylor ;
— celui de $\frac{1}{(n+1)!} f^{(n+1)}([a,b])$.

Pour ce faire, l'idée est de :

1. commencer par identifier un ensemble de fonctions appelées *fonctions de base*, pour lesquelles ces deux tâches sont aisées : $\sin, \cos, \exp, \tan, \log, x^{p/q}$ et de façon plus générale, les fonctions D-finies [139] (pour différentiellement finies) ou holonomes, soit les solutions d'équations différentielles linéaires à coefficients polynomiaux.
2. Ensuite, toute fonction doit être décomposée comme un arbre, ou un graphe acyclique dirigé (GAD), formé de ces fonctions de base et des opérations algébriques $+, -, \times, /, \circ$. Il faut alors définir une arithmétique spécifique associée à ces opérations algébriques : elles doivent opérer sur des approximations polynomiales rigoureuses, c.-à-d. des couples (P, Δ) où P est un polynôme et Δ un intervalle.

L'approche est donc la suivante :

— pour une fonction de base, on utilise le lemme 6.12. Pour une fonction de base D-finie, on peut par exemple utiliser les récurrences linéaires satisfaites par les coefficients pour les calculer et l'équation différentielle pour calculer un intervalle contenant le reste.
— pour une fonction f résultant d'une suite d'opérations $+, -, \times, /, \circ$ sur des fonctions de base (nous dirons de ces fonctions qu'elles sont composées) :
  1. on calcule d'abord une approximation polynomiale rigoureuse (P, Δ) pour chaque fonction de base de l'arbre décrivant la fonction f,
  2. puis on applique les règles arithmétiques spécifiques à ce type d'approximations polynomiales rigoureuses .

Définition 6.13. *On appelle modèle de Taylor une approximation polynomiale rigoureuse* $(P, \mathbf{\Delta})$ *calculée par cette approche.*

6.3.1 Remplacer les polynômes de Taylor par des approximations plus fines

Si les développements de Taylor donnent d'excellents approximants locaux, ils sont dépassés par d'autres classes d'approximation si l'on est en quête d'une très bonne approximation uniforme sur un intervalle. Ils peuvent aussi poser des problèmes si la fonction à approcher possède des singularités près de l'intervalle considéré. L'idée de les remplacer par une des approximations optimales ou quasi-optimales que nous avons présentées dans le chapitre 3, à savoir l'approximation minimax, la série de Tchebychev tronquée ou les interpolants de Tchebychev (rappelons qu'il s'agit d'un polynôme d'interpolation aux nœuds de Tchebychev (3.2) ou (3.3)) est donc survenue naturellement. Les séries tronquées de Tchebychev sont par exemple évoquées comme outils pour le calcul rigoureux dans les travaux [59, 60, 105]. Les auteurs parlent alors d'*ultra-arithmetic* pour qualifier le calcul d'approximations polynomiales rigoureuses à partir de séries de Fourier généralisées.

Comme nous l'avons vu dans le chapitre 3, on dispose de procédés de calcul efficace des interpolants de Tchebychev. De plus, si les fonctions sont D-finies, les articles [167, 13] donnent des procédés de calcul à la fois efficaces et stables numériquement des séries tronquées de Tchebychev.

L'algorithme de Remez, cf. l'algorithme 6 du chapitre 3, produit aussi efficacement des approximations minimax mais il est plus coûteux que les deux procédés précédents.

Quant aux restes, le point 6 du théorème 3.42 nous fournit un analogue du reste de la formule de Taylor-Lagrange : on se place sans perte de généralité sur $[-1, 1]$, soient P_n^* le polynôme minimax de degré au plus n de f, $(f_n)_{n\in\mathbb{N}}$ la suite de ses séries tronquées de Tchebychev et $(p_n)_{n\in\mathbb{N}}$ la suite des interpolants de Tchebychev de f, on a

$$\|f - P_n^*\|_\infty, \|f - f_n\|_\infty, \|f - p_n\|_\infty \in \frac{|f^{(n+1)}([-1,1])|}{2^n(n+1)!}. \tag{6.1}$$

Nous constatons donc que, du fait de l'évaluation par intervalles, l'encadrement du reste donné par ces trois formules est le même. Comme le coût de calcul du minimax est plus important, nous abandonnons cette alternative.

Pour le calcul de $\frac{|f^{(n+1)}([-1,1])|}{2^n(n+1)!}$, on dispose de plusieurs possibilités. Si la fonction est une fonction de base (voir plus haut), le reste est estimé de la même façon que pour les modèles de Taylor. Si, par exemple, on a $f = u \circ v$ alors :

— la méthode des différences divisées peut parfois donner des surestimations grossières, tout comme le calcul symbolique ;

— la différentiation automatique [79, 78, 12, 177, 153] est une technique bien connue pour calculer des dérivées d'ordre élevé en un point. Ici, on souhaite l'appliquer à des intervalles. Dans ce cadre, elle retourne un intervalle $\boldsymbol{K}$, contenant le reste, qui est obtenu en effectuant des opérations récursives sur des intervalles contenant $u^{(i)}([-1,1]), i = 0, \ldots, n+1$ et $v^{(i)}(\boldsymbol{J}), i = 0, \ldots, n+1$, où $v([-1,1]) \subset \boldsymbol{J}$. Cette méthode est meilleure que les deux précédentes mais peut malheureusement donner parfois, elle aussi, de très grandes surestimations du reste [33].

Pour compléter le dernier point, précisons que les modèles de Taylor donnent généralement une borne plus fine que celle calculée directement, à l'aide de la différentiation automatique par exemple [33, 142].

Vers des modèles de Tchebychev

Après ces échecs nous reste la stratégie en deux étapes déjà utilisée pour les modèles de Taylor. Une fois la fonction décomposée sous forme d'un arbre (ou un GAD) :

1. on calcule des modèles pour les fonctions de base,
2. on les combine à l'aide d'une arithmétique spécifique à ces modèles.

Nous savons traiter le premier point de façon performante dans les cas des interpolants de Tchebychev et des séries tronquées de Tchebychev. Notons que les restes (6.1) apportent une amélioration d'un facteur 2^n par rapport aux modèles de Taylor des fonctions de base.

Quant au second point, il faut développer cette arithmétique sur ces approximations polynomiales rigoureuses. Nous allons nous y atteler maintenant. À notre connaissance, cette arithmétique n'existait pas avant notre travail et ce dernier a constitué la première alternative performante aux modèles de Taylor.

Les modèles de Tchebychev sont implémentés en arithmétique multiprecision afin de garantir des calculs rigoureux. Néanmoins, afin de faciliter la lecture, nous les présentons en arithmétique exacte.

6.4 Modèles de Tchebychev

Nous introduisons donc la notion de modèle de Tchebychev de degré n pour une fonction f sur un intervalle $\boldsymbol{I}$ comme un couple $(P, \boldsymbol{\Delta})$ tel que :

— P est un polynôme de degré n, exprimé dans la base de Tchebychev, et qui est relié, soit à un interpolant de Tchebychev pour f, soit à une série tronquée de Tchebychev (cela dépend de l'approximation polynomiale choisie pour l'étape 1, celle des fonctions de base),

— $\boldsymbol{\Delta}$ est un intervalle contenant les valeurs du reste de $f - P$,

— pour les fonctions de base f, ce polynôme P coïncide soit avec un interpolant de Tchebychev pour f, soit avec la série tronquée de Tchebychev de f, et le calcul de $\boldsymbol{\Delta}$ est détaillé dans la sous-section 6.4.2.

— pour les fonctions composées, nous utilisons des règles algébriques définies dans les sous-section 6.4.3 et 6.4.4, de manière analogue à l'arithmétique des modèles de Taylor.

Notre cadre pratique est celui de l'arithmétique d'intervalles multiprécision et tous les algorithmes sont adaptés et implémentés pour cela.

Nous présentons maintenant des opérations de base qui seront utilisées par les opérations sur les modèles.

6.4.1 Opérations sur les polynômes dans la base de Tchebychev

Sur $\boldsymbol{I} = [a, b]$, on définit les polynômes de Tchebychev de première espèce sur $\boldsymbol{I}$ par $T_n^{[a,b]}(x) = T_n\left(\frac{2x-b-a}{b-a}\right), n \geqslant 0$. Nous utilisons les analogues, dans le cas de l'intervalle $[a, b]$, des définitions (3.2)

et (3.3) et du théorème 3.42.

Pour tout $n \geqslant 0$, $T_{n+1}^{[a,b]}$ possède $n+1$ racines réelles distinctes dans $[a,b]$, appelées nœuds de Tchebychev de première espèce :

$$\mu_k^{[a,b]} = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{(k+1/2)\,\pi}{n+1}\right),\ k = 0, \ldots, n. \tag{6.2}$$

Les extrema de T_n sont les nœuds de Tchebychev de seconde espèce :

$$\nu_k^{[a,b]} = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{k\pi}{n}\right),\ k = 0, \ldots, n. \tag{6.3}$$

L'analogue du point 6 du théorème 3.42 nous donne les formules que nous allons utiliser pour le calcul du reste de l'approximation :

Théorème 6.14. *Soit $f \in \mathcal{C}^{n+1}([-1,1])$. On note $(f_n)_{n\in\mathbb{N}}$ la suite de ses séries tronquées de Tchebychev et $(p_n)_{n\in\mathbb{N}}$ la suite des interpolants de Tchebychev de f (en utilisant les nœuds de Tchebychev de 1ère ou de 2nde espèce), il existe ξ_1 et $\xi_2 \in (a,b)$ tels que*

$$\|f - f_n\|_\infty = \frac{(b-a)^{n+1}|f^{(n+1)}(\xi_1)|}{2^{2n+1}(n+1)!},\ \textit{voir [58]}; \tag{6.4}$$

$$\|f - p_n\|_\infty = \frac{(b-a)^{n+1}|f^{(n+1)}(\xi_2)|}{2^{2n+1}(n+1)!},\ \textit{cf. théorème 3.18 et proposition 3.19}. \tag{6.5}$$

On considère deux polynômes de degré n exprimés dans la base de Tchebychev : $P(x) = \sum_{i=0}^{n} p_i T_i^{[a,b]}(x)$ et $Q(x) = \sum_{i=0}^{n} q_i T_i^{[a,b]}(x)$.

Addition. On a évidemment $P(x) + Q(x) = \sum_{i=0}^{n} (p_i + q_i) T_i^{[a,b]}(x)$, ce qui prend $O(n)$ opérations.

Multiplication. Le produit $P(x) \cdot Q(x) = \sum_{k=0}^{2n} c_k T_k^{[a,b]}(x)$, avec $c_k = (\sum_{|i-j|=k} p_i \cdot q_j + \sum_{i+j=k} p_i \cdot q_j)/2$. Cela provient [146] de l'identité $T_i^{[a,b]}(x) \cdot T_j^{[a,b]}(x) = (T_{i+j}^{[a,b]} + T_{|i-j|}^{[a,b]})/2$. Le coût s'élève à $O(n^2)$ opérations. Noter qu'il existe un produit rapide en $O(n \log n)$, à base de FFT, des polynômes de Tchebychev [9, 23, 73, 174] mais à ce jour, on ne sait toujours pas le réaliser avec une certification fine des résultats.

Borne sur l'image du polynôme. Ici, nous nous servons tout simplement du fait que les polynômes de Tchebychev prennent leur valeur dans $[-1,1]$. On obtient donc, en $O(n)$ opérations, la propriété : $\forall x \in [a,b], P(x) \in p_0 + \sum_{i=1}^{n} p_i \cdot [-1,1]$.

6.4.2 Fonctions de base

Pour nous, les fonctions de base sont les fonctions trigonométriques, exponentielle, logarithme, $x \mapsto 1/x$, puissance ou tout autre fonction dont on peut exploiter des propriétés spécifiques pour obtenir facilement un modèle de Tchebychev. Ce sont donc les fonctions D-finies ici encore. Pour toutes ces fonctions, nous calculons directement un modèle $(P, \mathbf{\Delta})$ formé d'un interpolant de Tchebychev P ou d'une série tronquée de Tchebychev et un intervalle $\mathbf{\Delta}$ bornant le reste.

Calcul du polynôme d'interpolation

Nous utilisons les formules de la proposition 3.21. Cela donne $P(x) = \sum_{i=0}^{n} p_i T_i^{[a,b]}(x)$, avec $p_i = \sum_{k=0}^{n} \frac{2}{n+1} f\left(\mu_k^{[a,b]}\right) T_i^{[a,b]}\left(\mu_k^{[a,b]}\right)$ si l'on utilise les nœuds de Tchebychev de première espèce ou $P(x) =$

$\sum_{i=0}^{n} p_i T_i^{[a,b]}(x)$, avec $p_i = \sum_{k=0}^{n} \frac{2}{n} f\left(\nu_k^{[a,b]}\right) T_i^{[a,b]}\left(\nu_k^{[a,b]}\right)$ si l'on utilise les nœuds de Tchebychev de seconde espèce.

Le coût de calcul s'élève à $O(n^2)$ opérations. Comme nous l'avons rappelé dans le chapitre 3, l'utilisation de la FFT peut diminuer ce coût à $O(n \log n)$ [173]. Cependant, le problème de certification fine de l'arithmétique rapide se pose de nouveau et nous sommes donc conduits à privilégier la méthode quadratique.

Calcul de la série tronquée de Tchebychev

Nous utilisons la méthode présentée dans [13] ou encore celle donnée dans [167].

Estimation du reste

Nous calculons un encadrement du reste $f - P$ sur $[a, b]$ en utilisant le théorème 6.14. Cela se réduit à borner $f^{(n+1)}$ sur $[a, b]$, ce que l'on fait en utilisant l'équation différentielle, de manière analogue à ce que nous avons décrit pour les modéles de Taylor dans la sous-section 6.3.

Remarque 6.15. En pratique, si la fonction $f^{(n+1)}$ est monotone sur $[a, b]$ (cela peut sembler un critère bizarre, mais on peut le tester facilement sur les fonctions de base communes), on peut obtenir une estimation optimale. Cela renforce, parfois significativement, l'approche. Notre lectrice peut consulter **C-12** ou [98] pour plus de détails.

Borner la partie polynomiale

Quand on utilise cette approche, on a besoin de calculer des bornes pour l'image des polynômes impliqués dans les modèles. Nous notons $B(P)$ un intervalle (il n'est pas unique) contenant l'image d'un polynôme P sur un intervalle $\boldsymbol{I}$. Plusieurs méthodes existent et en général, il faut choisir un certain compromis entre la vitesse et la finesse des bornes. Pour les modèles de Taylor, la méthode la plus rapide (et grossière) est une évaluation par intervalles à la Horner. Des procédés plus complexes existent et donnent en général des bornes plus fines : les méthodes LDB, QDB [15] ou l'utilisation de la conversion dans la base des polynômes de Bernstein [157, 222]. Pour les polynômes univariés, des techniques plus lentes fondées sur l'isolation de racines peuvent être utilisées pour fournir des bornes très fines [184].

Dans notre cas, nous nous sommes concentrés sur la vitesse et donc, pour borner l'image d'un polynôme dans la base de Tchebychev, nous utilisons la méthode immédiate décrite dans la section 6.4.1. Tout comme pour les modèles de Taylor, si l'on accepte une baisse de vitesse, des algorithmes plus fins peuvent être utilisés. Toutefois, la technique sommaire que nous utilisons se révèle très performante dans la plupart des exemples traités jusqu'à présent.

6.4.3 Addition et multiplication

Dans ce qui suit, nous considérons deux modèles de Tchebychev relatifs à deux fonctions f_1 et f_2, sur $\boldsymbol{I}$, de degré n : $(P_1, \boldsymbol{\Delta}_1)$ et $(P_2, \boldsymbol{\Delta}_2)$.

L'addition de deux modèles se fait tout simplement en additionnant les deux polynômes et les deux intervalles-reste :

$$(P_1, \boldsymbol{\Delta}_1) + (P_2, \boldsymbol{\Delta}_2) = (P_1 + P_2, \boldsymbol{\Delta}_1 + \boldsymbol{\Delta}_2).$$

Cela se fait en complexité linéaire, cf. section 6.4.1.

Quant à la correction, notons que : $\forall x \in \boldsymbol{I}, \exists \delta_1 \in \boldsymbol{\Delta}_1$ et $\delta_2 \in \boldsymbol{\Delta}_2$ tels que $f_1(x) - P_1(x) = \delta_1$ et $f_2(x) - P_2(x) = \delta_2$. D'où $f_1(x) + f_2(x) - (P_1(x) + P_2(x)) = \delta_1 + \delta_2 \in \boldsymbol{\Delta}_1 + \boldsymbol{\Delta}_2$.

Pour la multiplication, nous avons

$$f_1(x) \cdot f_2(x) = P_1(x) \cdot P_2(x) + P_2(x) \cdot \delta_1 + P_1(x) \cdot \delta_2 + \delta_1 \cdot \delta_2.$$

Nous observons que $P_1 \cdot P_2$ est un polynôme de degré $2n$. On développe $P_1 \cdot P_2$ dans la base de Tchebychev. On note $(P_1 \cdot P_2)_{0 \ldots n}$ la composante de $P_1 \cdot P_2$ suivant la base $\{T_0^{[a,b]}, \ldots, T_n^{[a,b]}\}$ et $(P_1 \cdot P_2)_{n+1 \ldots 2n}$ la partie restante, c.-à-d. $P_1 \cdot P_2$ - $(P_1 \cdot P_2)_{0 \ldots n}$.

On obtient un modèle de Tchebychev pour $f_1 \cdot f_2$ en considérant un intervalle

$$\boldsymbol{\Delta} = B((P_1 \cdot P_2)_{n+1\ldots 2n}) + B(P_2) \cdot \boldsymbol{\Delta}_1 + B(P_1) \cdot \boldsymbol{\Delta}_2 + \boldsymbol{\Delta}_1 \cdot \boldsymbol{\Delta}_2.$$

Les bornes pour les polynômes peuvent se calculer comme suggéré dans la sous-section 6.4.2.

Dans notre cadre, cf. section 6.4.1, le nombre d'opérations nécessaires pour multiplier deux modèles est $O(n^2)$.

6.4.4 Composition

Quand on a besoin d'un modèle pour $f_1 \circ f_2$, la première chose à faire est la prise en compte du fait que l'image de f_2 doit être incluse dans l'ensemble de définition de f_1. On commence donc par calculer $B(P_2) + \boldsymbol{\Delta_2}$. Si l'inclusion est vérifiée, on a alors

$$(f_1 \circ f_2)(x) - P_1(f_2(x)) \in \boldsymbol{\Delta_1} \tag{6.6}$$

qui entraîne

$$(f_1 \circ f_2)(x) \in P_1(P_2(x) + \boldsymbol{\Delta_2}) + \boldsymbol{\Delta_1}.$$

Dans cette formule, les seuls coefficients polynomiaux et les seuls restes impliqués sont ceux des modèles de Tchebychev de f_1 et f_2 qui sont des fonctions de base. Comme nous l'avons vu précédemment, on dispose de formules simples pour calculer les coefficients et les restes de ces fonctions. Cependant, quand on utilise la formule (6.6), il n'est pas évident d'en extraire un polynôme et une borne sur le reste. Guidés par la méthode de composition des modèles de Taylor [142, 222], on souhaite réduire ce processus d'extraction à des multiplications et des additions de modèles de Tchebychev.

Dans notre cas, la différence est que P_1 et P_2 sont des polynômes représentés dans la base de Tchebychev, et non dans la base monomiale. Il n'est pas souhaitable de procéder à des conversions de la base de Tchebychev à la base monomiale, d'effectuer des calculs identiques à ceux faits sur les modèles de Taylor et de revenir à la base de Tchebychev : il y a un risque d'instabilité numérique des conversions, ce qui pourrait dégrader la qualité de nos restes. Par conséquent, le calcul de la composition de deux modèles de Tchebychev requiert un nouvel algorithme.

Comme l'algorithme de composition des modèles de Taylor s'appuie sur une adaptation du schéma d'évaluation de Horner [156, Algorithme 6.3], il est naturel d'essayer d'utiliser une adaptation du schéma d'évaluation de Clenshaw (algorithme 7), le pendant du schéma de Horner pour la base de Tchebychev. Dans notre cas, la variable x en laquelle la somme est évaluée est un modèle de Tchebychev et les multiplications et les additions sont des opérations sur des modèles de Tchebychev. De plus, cet algorithme nécessite une nombre linéaire d'opérations sur les modèles.

Algorithme 11 Composition de modèles de Tchebychev dans la base de Tchebychev : « $(P_1, \boldsymbol{\Delta_1}) \circ (P_2, \boldsymbol{\Delta_2})$ »

Entrée : Deux modèles de Tchebychev $(P_1, \boldsymbol{\Delta_1})$ et $\circ(P_2, \boldsymbol{\Delta_2})$, représentant respectivement les fonctions f_1 et f_2

Sortie : Un modèle de Tchebychev pour $f_1 \circ f_2$

// *modèles de Tchebychev pour 0*

1: $(C_{n+2}, \boldsymbol{R_{n+2}}) \leftarrow (0, [0, 0])$

2: $(C_{n+1}, \boldsymbol{R_{n+1}}) \leftarrow (0, [0, 0])$

// On note $P_{1,j}$ le j-ème coefficient de P_1

3: **pour** $j = n, \ldots, 1$ **faire**

4: $\quad (C_j, \boldsymbol{R_j}) \leftarrow 2 \cdot (P_2, \boldsymbol{\Delta_2}) \cdot (C_{j+1}, \boldsymbol{R_{j+1}}) - (C_{j+2}, \boldsymbol{R_{j+2}}) + (P_{1,j}, [0, 0])$;

5: **fin pour**

6: $(C, \boldsymbol{R}) \leftarrow (P_2, \boldsymbol{\Delta_2}) \cdot (C_1, \boldsymbol{R_1}) - (C_2, \boldsymbol{R_2}) + (P_{1,0}, [0, 0])$

7: Renvoyer $(C, \boldsymbol{R} + \boldsymbol{\Delta_1})$

6.4.5 Croissance des coefficients et surestimation

La surestimation croît modérément durant l'exécution des opérations algébriques sur les modèles de Tchebychev. Cela est dû aux propriétés de convergence très favorables, lorsque la fonction est suffisamment régulière, énoncées dans le théorème 3.42. Tout comme pour les modèles de Taylor, quand on compose deux modèles de Tchebychev (de fonctions assez régulières), les bornes des intervalles contribuant au reste final rapetissent pour les coefficients des polynômes de Tchebychev de plus haut degré, ce qui produit une surestimation réduite du reste final.

6.5 Résultats expérimentaux

Nous avons implémenté notre approche dans l'outil `Sollya` et dans un module Maple nommé ChebModels, en utilisant le module IntpakX[3]. Notre code Maple est disponible à l'adresse `http://www.ens-lyon.fr/LIP/Arenaire/Ware/ChebModels`. La table 6.1 montre la qualité des bornes sur l'erreur absolue que nous obtenons à l'aide des méthodes : modèles de Tchebychev, interpolant de Tchebychev combiné avec la différentiation automatique [34] (on désignera par ITDA cette approche dans cette section), et modèles de Taylor. Chaque ligne traite un exemple formé d'une fonction f, un intervalle $\boldsymbol{I}$ et le degré n de l'approximation polynomiale. Ces spécifications sont données dans la deuxième colonne.

Nous avons calculé un modèle de Tchebychev $= (P, \boldsymbol{\Delta})$ et nous affichons dans la colonne 3 un majorant rigoureux de $\max(|\alpha|, |\beta|)$ si $\boldsymbol{\Delta} = [\alpha, \beta]$.

Afin d'examiner la surestimation lors du calcul du modèle de Tchebychev, nous avons calculé la norme sup exacte $\varepsilon = \sup_{x \in I}\{|f(x) - P(x)|\}$ et l'affichons dans la quatrième colonne.

Nous donnons dans les colonnes 5 et 6 le reste calculé et l'erreur exacte obtenus quand on utilise directement un polynôme d'interpolation (directement signifie que le reste est calculé en utilisant les formules du théorème 6.14 et la différentiation automatique, suivant l'approche ITDA développée dans [34]). Enfin, nous présentons une borne pour le reste calculé à l'aide d'un modèle de Taylor et l'erreur exacte correspondante.

Les cinq premiers exemples proviennent d'une analyse faite dans [33] pour comparer les bornes obtenues par l'approche ITDA vs. les modèles de Taylor. Comme on le voit, les résultats exposés dans les quatre dernières colonnes montrent des résultats en défaveur de l'approche ITDA dès que les fonctions sont composées. En revanche les modèles de Tchebychev apportent des résultats supérieurs dans toutes les situations.

Le premier exemple présente une fonction de base qui est entière. Il n'y a presque pas de surestimation dans ce cas, quelque soit la méthode utilisée. Le deuxième exemple est aussi une fonction de base. Elle a des singularités dans le plan complexe (en $\pi/2 + \mathbb{Z}\pi$), mais l'intervalle $\boldsymbol{I}$ est relativement éloigné de ces singularités. Les trois méthodes présent une surestimation modeste. Le troisième exemple porte sur la même fonction, considérée sur un intervalle plus grand, et plus proche des singularités. Dans ce cas, les polynômes de Taylor ne sont plus de très bonnes approximations. Les quatrième et cinquième exemples sont des fonctions composées sur des intervalles plus grands. La surestimation due à la méthode ITDA la rend prohibitive, alors qu'elle reste raisonnable pour les modèles de Taylor et les modèles de Tchebychev. Le résultat retourné par l'approche modèle de Tchebychev permet de traiter le problème 6.1 de manière satisfaisante : si l'on remplace $e^{1/\cos(x)}$ par son modèle de Tchebychev (P, Δ), il ne nous reste plus qu'à estimer finement la norme sup du polynôme $P - P^*$ sur $[0, 1]$ pour obtenir une majoration très fine de $\|e^{1/\cos(x)} - P^*(x)\|_\infty$.

L'exemple 6 traite une fonction associée au phénomène de Runge, cf. section 3.2. Elle met en évidence la robustesse des modèles de Tchebychev par rapport aux deux autres méthodes. La méthode ITDA souffre d'une énorme surestimation du reste, imputable à l'utilisation de la différentiation automatique. Quant au modèle de Taylor, le problème vient de la présence des singularités $\pm i/2$: elles sont situées à l'intérieur du disque unité et empêchent donc le développement de Taylor de converger.

Pour terminer, revenons au problème 6.2, la question de quadrature certifiée de l'introduction. La conjugaison de méthodes adaptatives (ici, la détermination efficace d'une subdivision idoine de

3. `http://www.math.uni-wuppertal.de/~xsc/software/intpakX/`

No	$f(x), \boldsymbol{I}, n$	Modèles de Tchebychev		Interpolation + diff. automatique		Modèles de Taylor	
		Borne mod. Tchebychev	Borne exacte	Borne interp.	Borne exacte	Borne mod. Taylor	Borne exacte
1	$\sin(x), [3,\ 4], 10$	$\mathbf{1.19 \cdot 10^{-14}}$	$1.13 \cdot 10^{-14}$	$\mathbf{1.19 \cdot 10^{-14}}$	$1.13 \cdot 10^{-14}$	$1.22 \cdot 10^{-11}$	$1.16 \cdot 10^{-11}$
2	$\arctan(x), [-0.25,\ 0.25], 15$	$\mathbf{7.89 \cdot 10^{-15}}$	$7.95 \cdot 10^{-17}$	$\mathbf{7.89 \cdot 10^{-15}}$	$7.95 \cdot 10^{-17}$	$2.58 \cdot 10^{-10}$	$3.24 \cdot 10^{-12}$
3	$\arctan(x), [-0.9,\ 0.9], 15$	$\mathbf{5.10 \cdot 10^{-3}}$	$1.76 \cdot 10^{-8}$	$\mathbf{5.10 \cdot 10^{-3}}$	$1.76 \cdot 10^{-8}$	$1.67 \cdot 10^{2}$	$5.70 \cdot 10^{-3}$
4	$\exp(1/\cos(x)), [0,\ 1], 14$	$\mathbf{5.22 \cdot 10^{-7}}$	$4.95 \cdot 10^{-7}$	0.11	$6.10 \cdot 10^{-7}$	$9.06 \cdot 10^{-3}$	$2.59 \cdot 10^{-3}$
5	$\frac{\exp(x)}{\log(2+x)\ \cos(x)}, [0,\ 1], 15$	$\mathbf{9.11 \cdot 10^{-9}}$	$2.21 \cdot 10^{-9}$	0.18	$2.68 \cdot 10^{-9}$	$1.18 \cdot 10^{-3}$	$3.38 \cdot 10^{-5}$
6	$\frac{1}{1+4x^2}, [-1, 1], 10$	$\mathbf{1.13 \cdot 10^{-2}}$	$6.17 \cdot 10^{-3}$	$1.50 \cdot 10^{7}$	$4.95 \cdot 10^{-3}$	$+\infty$	$8.20 \cdot 10^{2}$

TABLE 6.1 – Exemples de bornes obtenues par différentes méthodes

l'intervalle d'intégration) et du calcul de modèles de Tchebychev pour l'intégrande $x \in [0,3] \mapsto \sin\left(\frac{1}{(10^{-3}+(1-x)^2)^{3/2}}\right)$ sur chaque sous-intervalle nous ont permis de garantir que la valeur de l'intégrale appartient à $0.749974368527[1;3]$.

Les méthodes que nous venons de présenter sont disponibles dans l'outil `Sollya`[4].

6.6 Approximations polynomiales rigoureuses : preuve formelle

La simplicité et la souplesse des techniques sollicitées dans **C-12** et [98] nous ont amenés à tenter de prouver formellement, avec l'assistant COQ, l'approximation polynomiale rigoureuse de fonctions. J'ai profité du cadre offert par l'action TaMaDi de l'ANR pour lancer un groupe de travail que j'ai nommé CoqApprox.

Nous y avons discuté d'algorithmes mêlant calculs symbolique et numérique pour l'approximation polynomiale de fonctions univariées et surtout de leur implémentation et de leur formalisation en COQ. Les questions connexes (entre autres : celles relatives à l'évaluation de fonctions, comme par exemple l'optimisation globale avec les sommes de carrés, etc) ont été aussi examinées. Nous nous réunissions tous les six mois et les participants réguliers venaient de Lyon (G. Hanrot, M. Joldeş, É. Martin-Dorel, M. Mayero, M. Mezzarobba, J.-M. Muller, I. Pasca), de Saclay (G. Melquiond, V. Magron, A. Mahboubi, B. Werner) et de Sophia Antipolis (L. Rideau, L. Théry). Ce travail a donné lieu à l'article **C-14**.

Notre doctorant Florent Bréhard fera un stage d'un mois cet été au sein du National Institute of Aerospace et de la base de Langley (Virginie, USA) de la NASA. Il aura l'occasion de développer des preuves formelles d'outils d'approximation polynomiale rigoureuse pour l'aéronautique.

6.7 Perspectives

Florent Bréhard a débuté sa thèse en septembre 2016 sous la direction de M. Joldeş, D. Pous et moi-même et devrait consacrer une partie significative de son doctorat à ces questions d'approximations polynomiales rigoureuses.

6.7.1 Méthodes spectrales certifiées

Les méthodes spectrales sont des outils, introduits par B. Galerkin et Lanczos et auxquels S. Orszag [11, 76] a donné un impact fondamental, de résolution d'ordre élevé approchée d'équations différentielles. Les techniques que nous développons depuis plusieurs années nous font espérer la possibilité de la certification de la qualité d'approximation fournie par ces méthodes spectrales, ce qui constituerait une avancée significative dans ce domaine. Ce travail sera mené en collaboration avec F. Bréhard, M. Joldeş, M. Mezzarobba et bénéficiera d'interactions régulières avec B. Salvy (LIP).

Des travaux [13] d'A. Benoit (ancien doctorant de B. Salvy), M. Joldeş et M. Mezzarobba montrent que l'on peut obtenir des modèles de Tchebychev de fonctions D-finies en complexité linéaire en le degré de l'approximation polynomiale.

Dans **S-5**, F. Bréhard, M. Joldeş et moi-même donnons une nouvelle méthode de calcul de modèles de Tchebychev de solutions d'équations différentielles linéaires ordinaires. À cette fin, nous étudions

4. http://sollya.gforge.inria.fr/

des propriétés théoriques comme la compacité, la convergence, la possibilité d'inverser l'opérateur intégral linéaire associé à l'équation différentielle, puis les troncatures de ce dernier, dans un espace, analogue à l'espace de Wiener, des coefficients de la série de Tchebychev. Ensuite, nous analysons la structure de matrice quasi-bande de ces opérateurs : elle donne lieu à des algorithmes très efficaces, à la fois pour les solutions approchées des équations différentielles et pour le calcul rigoureux de l'erreur d'approximation.

Nous réfléchissons actuellement à une extension de l'ensemble de ces outils à des développements en n'importe quelle famille de polynômes orthogonaux de Gegenbauer, qui regroupe les familles les plus usitées (Tchebychev, Legendre, Hermite, Laguerre). Ce serait un résultat à la fois utile en pratique et significatif sur le plan théorique.

6.7.2 Arithmétique rapide pour les approximations polynomiales rigoureuses

Nos implantations actuelles souffrent de la lenteur des opérations arithmétiques que nous effectuons sur les nœuds (du GAD représentant la fonction) formés d'approximations polynomiales rigoureuses. Une question importante pour nous est de voir si les techniques d'arithmétique rapide [9, 23, 73, 174] (Karatsuba ou FFT pour le produit par exemple) peuvent être mobilisées avec succès dans notre cadre de calcul certifié.

Cela fera l'objet d'un travail avec F. Bréhard et M. Joldeş.

6.7.3 Validation formelle des approximations

Je voudrais aussi poursuivre le travail que j'avais lancé, en tant que coordinateur, au sein du groupe de travail CoqApprox afin de développer des outils de formalisation en Coq de toutes ces approximations polynomiales rigoureuses. Cela sera fait en collaboration avec M. Joldeş, D. Pous (LIP) et notre doctorant F. Bréhard.

Cette partie-là est très exploratoire : un de nos principaux objectifs est d'arriver à des vitesses de développement bien plus élevées qu'actuellement pour ces aspects de formalisation, ce qui constitue un défi majeur pour la communauté de développeurs Coq.

6.7.4 Cas des équations différentielles non-linéaires

Jusqu'ici, nous avons abordé le cas des équations différentielles linéaires et s'il reste beaucoup de travail à effectuer dans ce cadre-là, il est important de considérer dès à présent le cas de certaines équations différentielles non linéaires. Ainsi, avec F. Bréhard et M. Joldeş, nous comptons réfléchir à l'adaptation de nos méthodes au cadre non-linéaire, en commençant par exemple par [219].

Deuxième partie

En amont : algorithmes bas niveau en arithmétique des ordinateurs, approximation diophantienne

Chapitre 7

Algorithmes bas niveau en arithmétique des ordinateurs

Le succès de l'implémentation en machine de bibliothèques d'évaluation de fonctions ou de cryptosystèmes à clé publique repose de manière critique sur l'existence de primitives de calcul arithmétique optimisées. Je présente maintenant brièvement les travaux que j'ai réalisés dans ces domaines.

7.1 Fonctions calculables à l'aide d'une instruction fused multiply-and-add

L'instruction fused multiply-and-add (`fma`) associe à trois nombres flottants x, y, z l'arrondi correct de $xy + z$. Elle est disponible sur certains processeurs actuels tels que le Power PC d'IBM ou l'Itanium d'Intel/HP et permet d'accélerer certains calculs.

Ainsi, en collaboration avec J.-M. Muller et S. K. Raina (LIP) **J-3**, nous avons développé des techniques pour accélérer le calcul, avec arrondi correct, du rapport x/y quand y est connu avant x. C'est le cas lorsque y est une constante (une application évidente de ce cas de figure est l'algorithme du pivot de Gauss) ou lorsque y est connu avant la compilation. Les algorithmes fournis visent à réduire la latence de la division à une multiplication et un `fma`. Nous avons montré que cet objectif est réalisé si une précision interne plus grande (un bit supplémentaire suffit) est disponible (c'est le cas sur l'Itanium par exemple) ou si y satisfait à certaines conditions facilement vérifiables à la compilation.

Dans un travail commun avec J.-M. Muller **C-3** puis **J-9**, nous avions montré que le `fma` peut être utilisé pour effectuer très efficacement des multiplications, avec arrondi correct, par une constante C qui n'est pas exactement représentable en virgule flottante. À la suite de ce travail, F. de Dinechin (AriC puis CITI, INSA Lyon), J.-M. Muller et moi-même avons entrepris d'utiliser des modifications des méthodes développées pour concevoir sur des FPGA un générateur d'architecture pour la multiplication avec arrondi correct d'un nombre flottant par une constante réelle quelconque. Ces résultats ont été publiés dans **C-9**.

J'ai eu l'occasion de développer d'autres algorithmes de bas niveau utilisant l'instruction `fma`, s'appliquant cette fois-ci au calcul de la racine carrée ou de la norme euclidienne. Plus précisément, si l'on désigne par les termes d'algorithme avec précision augmentée un algorithme renvoyant, en précision virgule flottante p, un résultat sous la forme de la somme de deux nombres flottants, avec une erreur relative de l'ordre de 2^{-2p}, M. Joldeş, P. Kornerup (U. Odense, Danemark), É. Martin-Dorel, J.-M. Muller et moi avons élaboré **C-13** des algorithmes avec précision augmentée calculant la racine carrée et la norme euclidienne $\sqrt{x^2 + y^2}$. Ces algorithmes sont accompagnés d'analyses d'erreur fines.

7.2 Conversions binaire/décimal rapides et avec arrondi correct, comparaisons binaire/décimal

L'arithmétique décimale est sans doute vouée à être de plus en plus utilisée dans le futur. Certaines équipes, telles que celle d'IBM par exemple, se sont lancées dans la réalisation d'outils ad hoc pour pouvoir effectuer les opérations de base et l'évaluation des fonctions élémentaires, si possible avec arrondi correct. A contrario, une idée, proposée notamment par John Harrison [87], et soutenue par Intel, consiste à essayer de tirer parti de l'excellente qualité des outils développés jusqu'à présent en arithmétique binaire. Il faut pour cela réaliser des outils puissants et fiables de conversion d'un format dans l'autre ou de comparaison de nombres aux formats différents.

J.-M. Muller a réalisé que certains de nos travaux précédents, publiés dans **J-9**, pourraient être d'une aide significative pour produire de tels outils. Nous avons donc pu montrer, en collaboration avec M. Ercegovac (UCLA, USA), N. Louvet, É. Martin-Dorel et A. Panhaleux (AriC, LIP), dans **C-11** que cette approche de l'arithmétique décimale via les conversions est performante, fiable et précise.

En 2013, C. Lauter, M. Mezzarobba, J.-M. Muller et moi avons établi un algorithme qui permet de comparer rapidement un nombre flottant au format binary64 à un nombre flottant au format decimal64, en supposant que les nombres au format décimal sont encodés en binaire (l'encodage privilégié par INTEL) conformément au standard IEEE 754-2008. L'article **C-16** présentant ces résultats a été publié dans les actes de la conférence ARITH. Depuis, nous avons significativement généralisé dans **J-12** ce travail à l'ensemble des nombres flottants binaires et décimaux.

7.3 Algorithmes et implantations matérielles efficaces pour la cryptographie : cryptographie fondée sur les couplages

La cryptographie fondée sur les couplages sur les courbes algébriques est un domaine de recherche très actif depuis le début des années 2000. Elle a permis d'obtenir des réponses satisfaisantes à plusieurs problèmes ouverts relativement anciens en cryptographie, tels que le chiffrage fondé sur l'identité. Ces solutions étaient pour beaucoup seulement à l'état de protocoles et nécessitaient des implantations efficaces pour convaincre les utilisateurs potentiels, tels que les industriels, de l'intérêt pratique considérable qu'elles constituent. Les couplages considérés sont essentiellement les couplages de Tate et η_T (on peut consulter [71] et [8] pour des définitions et propriétés de ces objets) sur des courbes elliptiques et hyperelliptiques supersingulières, définies sur des corps finis de la forme $\mathbb{F}_{2^m}$ et $\mathbb{F}_{3^m}$ (on peut les considérer sur d'autres corps finis mais les performances, permettant de garantir un niveau de sécurité équivalent, sont alors bien plus faibles). Les microprocesseurs généralistes sont intrinsèquement mal adaptés aux calculs sur les corps finis de petite caractéristique et les implantations logicielles sont donc lentes. Il y a donc un besoin critique d'implantations matérielles performantes sur des processeurs dédiés, tels que les FPGA.

J'ai entamé en 2006 un travail, principalement avec J.-L. Beuchat (Université de Tsukuba, Japon), mais aussi E. Okamoto (Université de Tsukuba, Japon), M. Shirase (Future Université de Hakodate, Japon) et T. Takagi (Future Université de Hakodate), visant à obtenir des implantations optimisées du couplage η_T et du couplage de Tate dans le cas des caractéristiques 2 et 3. Nous avons été rapidement rejoints par J. Detrey (à l'époque doctorant au sein d'Arénaire et maintenant chargé de recherche INRIA dans l'équipe CARAMEL du LORIA).

Notre collaboration a donné d'abord lieu aux articles **C-6** et **C-7**, ce dernier ayant obtenu le prix du meilleur article de CHES, la conférence de référence dans le domaine de la cryptologie matérielle.

Les résultats précédents ont été repris et étendus dans l'article **J-10**. Nous y analysons plusieurs algorithmes de calcul du couplage η_T en caractéristique 3 et suggérons plusieurs améliorations. Ces algorithmes utilisent l'addition, la multiplication, l'élévation au cube, l'inversion et parfois les racines cubiques sur $\mathbb{F}_{3^m}$. Nous proposons un accélérateur matériel s'appuyant sur un opérateur arithmétique unifié capable d'effectuer toutes les opérations requises par un algorithme donné. Enfin, nous décrivons l'implantation d'un coprocesseur compact dans le cas où le corps fini considéré est $\mathbb{F}_{3^{97}}$, donné dans ce travail par $\mathbb{F}_3[x]/(x^{97} + x^{12} + 2)$, implantation qui offre les meilleures performances de la littérature.

Dans l'article **C-10** écrit avec J.-L. Beuchat, J. Detrey, F. Rodríguez-Henríquez (Instituto Politécnico Nacional, Mexique) et E. Okamoto nous avons proposé une étude approfondie du couplage de Tate modifié en caractéristiques 2 et 3. Nous considérons d'abord le couplage η_T introduit par Barreto *et al.* et nous détaillons diverses améliorations algorithmiques dans le cas de la caractéristique 2. Nous montrons ensuite comment le couplage de Tate modifié peut être calculé à partir du couplage η_T avec un surcoût quasiment nul. Enfin, nous avons mené une comparaison poussée des implantations matérielles de ces couplages pour les deux caractéristiques. Nos expériences semblent indiquer un avantage en faveur de la caractéristique trois. Nos implantations sont actuellement les plus performantes (notre mesure est le produit «temps par surface») connues.

En collaboration avec des chercheurs de l'équipe CACAO (aujourd'hui CARAMEL) du Loria (Nancy) et les chercheurs de Tsukuba et Hakodate déjà cités, nous avons obtenu un financement pour la période 2008-2009 dans le cadre d'un Partenariat Hubert Curien Sakura-Ayame (Égide-INRIA).

De plus, suite à un processus sélectif, j'ai obtenu en 2010 un financement NiCT (National Institute of Information and Communications) japonais pour un séjour d'un moins au L.C.I.S. de l'université de Tsukuba.

Chapitre 8

Approximation diophantienne

Cette partie concerne mon travail en théorie des nombres. Le point commun des quatre sujets que je vais présenter est le traitement effectif et/ou algorithmique que j'en fais. Les techniques de calcul formel utilisées dans 8.1 et surtout celles d'approximation mobilisées dans 8.2 se sont montrées fécondes dans les questions d'arithmétique des ordinateurs auxquelles j'ai réfléchi : je les ai directement utilisées ou elles m'ont inspiré pour les travaux **J-3, J-4, J-6, J-7, J-9, J-10, J-12, C-1, C-3, C-4, C-5, C-6, C-7, C-8, C-9, C-10, C-16, NC-1, S-3**.

8.1 Résolution effective de systèmes d'équations aux différences

Un polynôme exponentiel est une fonction entière (c.-à-d. holomorphe sur tout le plan complexe) de la forme $\sum_{s=1}^{S} H_{\delta_s}(z)e^{\delta_s z}$ avec H_{δ_s} dans $\mathbb{C}[z]$ et δ_s dans $\mathbb{C}$ pour tout s. Soit f une fonction entière solution du système d'équations aux différences

$$\begin{cases} \displaystyle\sum_{0\leqslant m\leqslant M} P_m(z)f(z+m\alpha) = \Phi(z), \\ \displaystyle\sum_{0\leqslant n\leqslant N} Q_n(z)f(z+n\beta) = \Psi(z), \end{cases} \tag{8.1}$$

où $(P_m)_{0\leqslant m\leqslant M}$ et $(Q_n)_{0\leqslant n\leqslant N}$ sont des suites finies de $\mathbb{C}[z]$ avec $P_M Q_N P_0 Q_0 \neq 0$, α et β deux nombres complexes $\mathbb{R}$-linéairement indépendants, et Φ et Ψ deux polynômes exponentiels. J.-P. Bézivin et F. Gramain ont montré dans [17] et [18] que f est nécessairement le quotient d'un polynôme exponentiel par un polynôme et que ce résultat reste vrai quand la seconde équation est remplacée par une équation différentielle dont les coefficients appartiennent à $\mathbb{C}[z]$. Dans un travail commun avec L. Habsieger (A2X, Univ. Bordeaux 1 puis Inst. C. Jordan, Univ. Lyon 1) **J-1**, nous avons généralisé ce résultat au cas $\alpha/\beta \in \mathbb{R}\backslash\mathbb{Q}$, nous avons proposé une nouvelle approche à ce problème qui nous a permis de retrouver partiellement des résultats de [17] et [18] et nous avons donné la preuve d'un algorithme inspiré par le chapitre 8 de [171], qui produit les solutions entières du système (8.1) quand $\Phi = \Psi = 0$. De plus, une légère modification de l'algorithme nous a permis de traiter aussi le cas d'un système où la seconde équation est une équation différentielle.

J'ai donné une version plus performante de cet algorithme dans **S-1**. Elle permet en outre de traiter le cas de systèmes où les seconds membres sont des polynômes exponentiels quelconques. Cet algorithme implanté en Maple est disponible sur la page `http://perso.ens-lyon.fr/nicolas.brisebarre/Pasrec/pasrec.html`

8.2 Détermination explicites de mesures d'irrationalité de certaines constantes

Soit y un nombre réel, on dit que le nombre réel μ est une mesure (effective) d'irrationalité de y s'il existe $C(y,\mu) \in {\mathbb{R}_+}^*$ (effectivement calculable) tel que

$$\forall (p,q) \in \mathbb{Z} \times \mathbb{N}^*, \quad \left| y - \frac{p}{q} \right| \geqslant \frac{C(y,\mu)}{q^\mu}. \tag{8.2}$$

On sait, par un théorème de Legendre, que, pour tout nombre y irrationnel, une mesure d'irrationalité de y est nécessairement supérieure ou égale à 2. L'objectif est de prouver que les constantes classiques (nombres algébriques irrationnels, $\pi, \log 2, \log 3, \ldots$) admettent $2+\varepsilon$, pour tout $\varepsilon > 0$, comme mesure d'irrationalité. On sait (voir le théorème 5.3) que c'est le cas pour presque tout réel y mais on n'est pas capable d'exhiber l'ensemble de mesure nulle en question. J'ai mené dans **J-2** une étude systématique de l'approximation par des rationnels de certaines valeurs de la fonction logarithme. Elle a consisté tout d'abord à déterminer certaines classes de fractions rationnelles reliées aux approximants de Padé de la fonction $f : z \mapsto \log(1-1/z)$ et à développer des procédures algorithmiques permettant d'établir des mesures d'irrationalité de certaines valeurs de f à partir de ces fractions rationnelles. J'ai implanté ces algorithmes en GP/PARI et j'ai effectué des calculs intensifs qui m'ont permis d'unifier des travaux antérieurs [185, 89, 55, 56, 35] et aussi d'infirmer une annonce de nouveau record faite dans [35].

8.3 Fonctions modulaires

L'invariant modulaire j est une fonction fondamentale en théorie des nombres [190], en particulier dans certains problèmes de cryptographie fondée sur les courbes elliptiques. On peut la définir par exemple comme suit :

$$j(\tau) = \frac{(1 + 240 \sum_{n \geqslant 1} \sigma_3(n) e^{2i\pi n\tau})^3}{e^{2i\pi\tau} \prod_{n \geqslant 1} (1 - e^{2i\pi n\tau})^{24}}, \text{ pour } \tau \in \mathfrak{H} = \{z \in \mathbb{C}; \Im(z) > 0\}, \tag{8.3}$$

avec $\sigma_3(n) = \sum_{d|n} d^3$. Cette fonction est notamment 1-périodique, ce qui, joint à son caractère holomorphe sur $\mathfrak{H}$, entraîne qu'elle est développable en série de Fourier. Ses coefficients de Fourier interviennent dans de nombreux problèmes et l'estimation précise et effective de leur taille est une question importante.

Dans un travail en collaboration avec G. Philibert (Univ. St-Étienne) **J-5**, nous avons établi des majorants et des minorants fins des coefficients de Fourier de j^m pour tout $m \in \mathbb{N} \setminus \{0\}$. Ces résultats améliorent tous ceux connus auparavant et notamment [141, 91].

8.4 Corps de séries formelles

On trouve une très bonne présentation de ce sujet dans [121, 122]. Soient un nombre premier p et un entier positif s, on considère le corps fini $\mathbb{F}_q$ à q éléments, l'anneau de polynômes $\mathbb{F}_q[T]$ et le corps de fractions rationnelles $\mathbb{F}_q(T)$. Nous utilisons la valeur absolue définie sur $\mathbb{F}_q(T)$ par $|P/Q| = |T|^{\deg P - \deg Q}$ pour tous $P, Q \in \mathbb{F}_q(T)$, $Q \neq 0$, où $|T|$ est un nombre réel fixé plus grand que 1. En complétant $\mathbb{F}_q(T)$ avec cette valeur absolue, on obtient un corps, le corps des séries formelles en $1/T$ et à coefficients dans $\mathbb{F}_q$ et que l'on notera $\mathbb{F}(q)$. Si α est un élément non nul de $\mathbb{F}(q)$, alors

$$\alpha = \sum_{k \leqslant k_0} a_k T^k,$$

avec $k_0 \in \mathbb{Z}, a_k \in \mathbb{F}_q, a_{k_0} \neq 0$ et $|\alpha| = |T|^{k_0}$. On s'intéresse à l'approximation rationnelle sur $\mathbb{F}(q)$ et on peut y définir un algorithme de calcul de fraction continue : les éléments de $\mathbb{F}_q[T]$ sont les analogues des

entiers relatifs, ceux de $\mathbb{F}_q(T)$ les analogues des nombres rationnels et ceux de $\mathbb{F}(q)$ les analogues de $\mathbb{R}$. À l'heure actuelle, dans le cadre de l'approximation diophantienne classique sur $\mathbb{R}$, on ne sait toujours pas dire grand chose du cas fondamental des nombres algébriques de degré $\geqslant 3$. Si les développements en fraction continue des nombres quadratiques sont bien compris, il n'en est rien des nombres algébriques de degré supérieur. On ne sait pas par exemple s'il en existe un avec des quotients partiels bornés ou s'il en existe un avec des quotients partiels non bornés. La situation est très différente dans le cas de $\mathbb{F}(q)$ qui dispose de riches résultats sur ces questions de fractions continues. De façon plus générale, on peut considérer le corps de séries formelles $\mathbb{F}(K)$, dans lequel K remplace $\mathbb{F}_q$. J'ai débuté une étude de ce domaine grâce à Alain Lasjaunias (IMB, Univ. Bordeaux I) et dans l'article **J-13**, en collaboration avec Bill Allombert (IMB, Univ. Bordeaux I) et Alain, nous avons exhibé un développement en fraction continue remarquable sur $\mathbb{F}(\mathbb{Q})$.

Chapitre 9

Une synthèse de mes perspectives de recherche

Afin de donner un peu plus de visibilité et de cohérence à mon projet scientifique, je vais maintenant regrouper les perspectives que j'ai présentées en fin des trois derniers chapitres. J'en profite pour les compléter et les présenter dans le cadre d'un projet global que j'ai intitulé « Vers une bibliothèque de calcul numérique à niveau de sûreté variable ».

Je souhaite consacrer mes cinq prochaines années de recherche à l'extension de mes travaux consacrés à l'approximation et à ses applications :

— approximation (certifiée) de fonctions et applications liées à ces approximants, telles que l'optimisation globale ou l'intégration de ces fonctions sur certains domaines,
— synthèse automatique de filtres (quasi-optimaux) en traitement du signal.

Les projets relatifs sont présentés dans les sections 9.1, 9.2 et 9.3.

Un aspect important du premier point est que nous nous efforçons de nous détacher progressivement de la nécessité d'avoir la fonction de départ donnée explicitement, comme somme d'une série de Taylor par exemple, et de travailler directement sur une donnée implicite de la fonction sous forme de solution d'une équation différentielle par exemple. C'est un pas important en pratique puisque la plupart du temps, les mathématiciens, physiciens, chimistes ou biologistes, les ingénieurs ont affaire à des équations ou des systèmes différentiels. Nous souhaitons qu'ils puissent les évaluer de manière efficace et certifiée.

Quant au second point, il nous semble important de fournir à la communauté de traitement de signal des outils de calcul robustes et à résultats garantis.

Dans les dix ans qui viennent, je souhaiterais coordonner un projet d'ampleur consistant à développer une bibliothèque de routines calculatoires, offrant trois niveaux de sûreté que l'on qualifie comme suit :

— numérique, i.e. la bibliothèque renvoie un résultat sous la forme d'un nombre flottant,
— certifiée, i.e. la bibliothèque renvoie un résultat sous la forme d'un nombre flottant et d'une erreur ε, aussi fine que possible, tels que le flottant retourné se trouve à une distance ε de la valeur mathématique du résultat,
— formalisé, i.e. le calcul est fait à l'aide d'un assistant de preuve tel que COQ ou certaines étapes-clés du calcul sont formalisables.

Notre espoir est que cette bibliothèque joue un rôle-clé pour, au moins, les deux types d'utilisateurs suivants : les mathématiciens élaborant des preuves assistées par ordinateur en analyse [82, 117, 88, 206] et les ingénieurs qui travaillent sur des systèmes critiques (avionique, spatial, etc.) pour lesquels la certification, voire la formalisation des calculs est cruciale. Je reviendrai sur ce projet de bibliothèque dans la section 9.4.

9.1 Approximation efficace en machine

L'absence d'outils efficaces pour attaquer directement CVP_∞ nous a amené à nous placer dans le cadre euclidien et considérer alors la question sous l'angle d'un CVP_2. Toutefois, l'article [43] propose un algorithme d'attaque directe du CVP_∞ que nous avons à résoudre. Je voudrais donc essayer de le mettre en œuvre et tester ses possibilités pratiques.

9.1.1 Synthèse de filtres RIF et RII

Je souhaite aussi continuer à creuser ces questions de qualité numérique en traitement du signal. Nous proposons actuellement une solution satisfaisante pour le cas des filtres RIF, allant jusqu'à la synthèse matérielle en FPGA de ces filtres. Il nous reste maintenant à avancer significativement sur le cas des filtres RII [6, 168, 176], dont la fonction de transfert est une fraction rationnelle, et plus un polynôme. La première difficulté est le développement de l'analogue de l'algorithme de Parks-McClellan. En effet, le calcul d'une fraction rationnelle de meilleure approximation est notoirement instable et il va nous falloir élaborer des méthodes robustes à cet effet. Peut-être aurons-nous à passer via des approximations légèrement sous-optimales mais qui nous permettront des calculs sûrs. La question de la quantification des coefficients nous posera de nouveaux problèmes puisqu'il faudra prendre en compte des contraintes additionnelles telles que la localisation des pôles afin d'assurer la stabilité du filtre.

S. Filip et moi-même projetons de proposer une bibliothèque C++ dont l'objectif sera d'être l'outil de référence pour la synthèse de filtres RIF et RII. À terme, elle devra être reversée dans la bibliothèque à trois niveaux de sécurité présentée dans ce projet.

9.2 Fonctions élémentaires et arrondi correct - Dilemme du fabricant de tables

En collaboration avec O. Robert, nous souhaitons approfondir l'étude de **J-14** présentée dans ce chapitre. Les résultats que nous avons obtenus restent partiels (je fais référence à la limitation $k \leqslant p/3$) et il y a un travail mathématique, a priori difficile, à faire sur certaines intégrales oscillantes [199] pour améliorer la situation.

J'ai évoqué dans la section 5.5 un travail en cours en collaboration avec Guillaume Hanrot qui propose une nouvelle approche algorithmique du dilemme du fabricant de tables pour les fonctions transcendantes **S-7**. Nous nous focalisons sur l'exemple central de la fonction exponentielle. Elle nous laisse entrevoir la possibilité de concevoir une bibliothèque mathématique, avec arrondi correct, performante pour le format binary128. C'était une tâche hors de portée jusqu'à présent. Outre la poursuite de la mise au point de cet article **S-7**, nous souhaiterions déployer à grande échelle le procédé qu'il expose.

9.3 Approximations polynomiales rigoureuses

Comme nous l'avons vu dans le chapitre 6, une façon de calculer des approximations polynomiales rigoureuses consiste à décomposer la fonction visée sous forme d'arbre (ou de DAG), puis de calculer des approximations polynomiales rigoureuses de chacun des nœuds, et enfin d'utiliser une arithmétique spécifique pour ces couples (P, Δ) où P est un polynôme généralisé et Δ un intervalle reste.

Florent Bréhard a débuté sa thèse en septembre sous la direction de M. Joldeş, D. Pous et moi-même et devrait consacrer une partie significative de son doctorat à ces questions d'approximations polynomiales rigoureuses.

9.3.1 Méthodes spectrales certifiées

Les méthodes spectrales sont des outils, introduits par B. Galerkin et Lanczos et auxquels S. Orszag[11, 76] a donné un impact fondamental, de résolution approchée d'équations différentielles d'ordre élevé. Les techniques que nous développons depuis plusieurs années nous font espérer la possibilité de la

certification de la qualité d'approximation fournie par ces méthodes spectrales, ce qui constituerait une avancée significative dans ce domaine. Ce travail sera mené en collaboration avec F. Bréhard, M. Joldeş, M. Mezzarobba et bénéficiera d'interactions régulières avec B. Salvy (LIP).

Des travaux [13] d'A. Benoit (ancien doctorant de B. Salvy), M. Joldeş et M. Mezzarobba montrent que l'on peut obtenir des modèles de Chebyshev en complexité linéaire en le degré de l'approximation polynomiale.

Dans **S-5**, F. Bréhard, M. Joldeş et moi-même donnons une nouvelle méthode de calcul de modèles de Tchebychev de solutions d'équations différentielles linéaires ordinaires. À cette fin, nous étudions des propriétés théoriques comme la compacité, la convergence, la possibilité d'inverser l'opérateur intégral linéaire associé à l'équation différentielle, puis les troncatures de ce dernier, dans un espace, analogue à l'espace de Wiener, des coefficients de la série de Tchebychev. Ensuite, nous analysons la structure de matrice quasi-bande de ces opérateurs : elle donne lieu à des algorithmes très efficaces, à la fois pour les solutions approchées des équations différentielles et pour le calcul rigoureux de l'erreur d'approximation.

Nous réfléchissons actuellement à une extension de l'ensemble de ces outils à des développements en n'importe quelle famille de polynômes orthogonaux de Gegenbauer, qui regroupe les familles les plus usitées (Chebyshev, Legendre, Hermite, Laguerre). Ce serait un résultat à la fois utile en pratique et significatif sur le plan théorique.

Un exemple d'application parlant (sur lequel nous reviendrons dans 9.3.5) est celui de la trajectoire d'un astéroïde. Supposons que celle-ci soit modélisée par une équation différentielle. Nous essayons alors de fournir une approximation polynomiale rigoureuse de la fonction, qui nous permettra de savoir de manière certifiée quelle est la zone dans laquelle risque de passer l'astéroïde. La forme de l'équation différentielle décide de la base de polynômes orthogonaux dans laquelle il faut calculer l'approximation polynomiale, d'où l'importance des techniques générales que nous essayons d'élaborer

9.3.2 Arithmétique rapide pour les approximations polynomiales rigoureuses

Nos implantations actuelles souffrent de la lenteur des opérations arithmétiques que nous effectuons sur les nœuds (du GAD représentant la fonction) formés d'approximations polynomiales rigoureuses. Une question importante pour nous est de voir si les techniques d'arithmétique rapide [9, 23, 73, 174] (Karatsuba ou FFT pour le produit par exemple) peuvent être mobilisées avec succès dans notre cadre.

Cela fera l'objet d'un travail avec F. Bréhard, M. Joldeş.

9.3.3 Validation formelle des approximations

Je voudrais aussi poursuivre le travail que j'avais lancé, en tant que coordinateur, au sein du groupe de travail CoqApprox afin de développer des outils de formalisation en Coq de toutes ces approximations polynomiales rigoureuses. Cela sera fait en collaboration avec M. Joldeş, D. Pous (LIP) et notre doctorant F. Bréhard.

Cette partie-là est très exploratoire : un de nos principaux objectifs est d'arriver à des vitesses de développement bien plus élevées qu'actuellement pour ces aspects de formalisation, ce qui constitue un défi majeur pour la communauté de développeurs Coq.

9.3.4 Cas des équations différentielles non-linéaires

Jusqu'ici, nous avons abordé le cas des équations différentielles linéaires et s'il reste beaucoup de travail à effectuer dans ce cadre-là, il est important de considérer dès à présent le cas de certaines équations différentielles non linéaires. Ainsi, avec F. Bréhard et M. Joldeş, nous comptons réfléchir à l'adaptation de nos méthodes au cadre non-linéaire, en commençant par exemple par [219].

9.3.5 Applications

Les approximations polynomiales rigoureuses sont des briques de base pour le calcul certifié. Nous déclinons maintenant des applications importantes que nous devrions pouvoir développer.

Quadrature certifiée

L'intégration numérique est une routine commune à tous les logiciels de calcul numérique ou symbolique tels que Maple, MATLAB, Mathematica, SAGE, etc. Aucun de ces outils n'offre de garantie sur le résultat renvoyé, comme nous l'avons illustré avec le problème 6.2.

Nos outils certifiés nous ont permis de conclure, cf. section 6.5, que, si Maple18 et SAGE donnent bien ici les dix premières décimales du résultat, la situation est globalement très insatisfaisante. La conjugaison de méthodes adaptatives (la détermination efficace d'une subdivision idoine de l'intervalle d'intégration dans le cas du problème 6.2) et du calcul d'approximations polynomiales rigoureuses des intégrandes sur chaque sous-intervalle devrait nous fournir des routines de quadrature certifiée satisfaisantes.

Preuve assistée par ordinateur : intégrales abéliennes en relation avec le 16ème problème de Hilbert

Nous espérons pouvoir nous appuyer sur ces routines pour tenter d'avancer sur un problème appelé 16ème problème de Hilbert *infinitésimal*, une version affaiblie de la seconde partie du 16ème problème de Hilbert [92], dont le but est de déterminer le nombre maximum et l'emplacement des cycles limites d'un champ de vecteurs polynomial plan de degré d. Ces cycles limites sont reliés à des intégrales abéliennes que l'on calcule actuellement à l'aide d'une méthode développée dans [97], méthode que nous souhaiterions significativement améliorer quant à son efficacité. Ce travail sera mené en collaboration avec F. Bréhard et W. Tucker.

Problématique des "Near-Earth Objects"

À terme, une très jolie question, qui m'avait été signalée par Mioara Joldeş, et que nous aimerions traiter le plus automatiquement possible, est celle des astéroïdes géocroiseurs (en anglais, on va parler plus généralement de near-Earth objects), dont l'orbite s'approche de celle de la Terre. L'étude [52] avait été publiée suite aux craintes soulevées par le passage de l'astéroïde Apophis (cf. `http://neo.jpl.nasa.gov/news/news146.html`).

Les astronomes effectuent des mesures qui aboutissent à une modélisation de la trajectoire de l'objet et nous souhaiterions être capables de les assister pour la suite de leur analyse en leur fournissant des approximations polynomiales rigoureuses de ces trajectoires. L'objectif est qu'ils puissent estimer le plus précisément possible les risques que pourraient faire courir le passage de l'objet près de notre planète.

9.4 Vers une bibliothèque de calcul à trois niveaux de sécurité

L'objectif à long terme est de proposer un ensemble de routines, en une variable pour commencer, pour calculer des valeurs, des intégrales, des zéros, des normes sup de fonctions solutions d'équations différentielles linéaires ou non. Deux questions se posent immédiatement : le mode de développement et la pérennité d'un tel outil. En ce qui concerne la pérennité, j'estime que cela fera partie de ma tâche de superviseur de ce projet. Quant au mode de développement, j'imagine un logiciel dont les routines seront écrites en C, C++ ou Coq et qui seront liées entre elle par un code Python. La possibilité de développer sera ouverte à tous et je vais essayer de l'organiser sous la forme suivante : mon idée est d'identifier des routines de calcul, à la fois intéressantes en elles-mêmes et susceptibles de jouer un rôle-clé dans des défis scientifiques tels que des questions ouvertes en ingénierie des systèmes critiques ou en théorie des systèmes dynamiques. Ces défis motiveront, je l'espère, un certain nombre de collègues à développer un code soigné pour ces routines.

Les deux premières classes de routines que je vise sont le calcul d'approximations polynomiales rigoureuses et leur versant formel, tels que présentés en 6.7.1 et 6.7.3, puis les routines de quadrature classiques (méthode de Gauss, méthode de Clenshaw-Curtis) et les variantes certifiées ou formalisables que nous aurons conçues, cf. 9.3.5.

Annexe A

Carrière et formation

Parcours professionnel

2007-$\cdots$ Chargé de recherche au CNRS.

2002-05 Délégation sur un poste de Chargé de Recherche INRIA dans le projet Arénaire du Laboratoire de l'Informatique du Parallélisme de l'ENS Lyon.

1999-2007 Maître de Conférences à l'Université J. Monnet de Saint-Étienne.

1998-99 ATER à l'Université Bordeaux 1.

1995-98 Allocataire de recherche à l'Université Bordeaux 1.

Cursus Universitaire

1995-98 Doctorat de Mathématiques Pures « Une étude de deux problèmes diophantiens » effectué sous la direction de Laurent Habsieger et soutenu le 25 septembre 1998 à l'Université Bordeaux 1.

1994-95 D.E.A. de Mathématiques Pures, Université Bordeaux 1.

1995 Agrégation externe de Mathématiques, option Mathématiques de l'Informatique.

Annexe B

Liste de publications

Toutes les publications de cette liste (excepté le livre) sont disponibles sur la page http://perso.ens-lyon.fr/nicolas.brisebarre/publi.html.fr.

Livres

B-1. J.-M. Muller, N. Brisebarre, F. de Dinechin, C.-P. Jeannerod, V. Lefèvre, G. Melquiond, N. Revol, D. Stehlé et S Torres. *Handbook of Floating-Point Arithmetic*. Birkhäuser Boston, 2010.

Journaux internationaux avec comité de lecture

J-1. N. Brisebarre et L. Habsieger. *Sur les fonctions entières à double pas récurrent*. Annales de l'Institut Fourier **49**-2, 653-671, 1999.

J-2. N. Brisebarre. *Irrationality measures of* $\log 2$ *and* $\pi/\sqrt{3}$. Experimental Mathematics **10**-1, 35-52, 2001.

J-3. N. Brisebarre, J.-M. Muller et S. K. Raina. *Accelerating correctly rounded floating-point division when the divisor is known in advance*. IEEE Transactions on Computers, vol. **53**, n. 8, 1069-1072, 2004.

J-4. N. Brisebarre, D. Defour, P. Kornerup, J.-M. Muller et N. Revol. *A new range reduction algorithm*. IEEE Transactions on Computers, vol **54**, n. 3, 331-339, 2005.

J-5. N. Brisebarre et G. Philibert. *Effective lower and upper bounds for the Fourier coefficients of powers of the modular invariant* j. Journal of the Ramanujan Mathematical Society, vol. **20**, n. 4, 255-282, 2005.

J-6. N. Brisebarre, J.-M. Muller et A. Tisserand. *Computing Machine-Efficient Polynomial Approximations*. ACM Transactions on Mathematical Software, vol. 32, n. 2, 236-256, Jun. 2006.

J-7. N. Brisebarre et J.-M. Muller. *Correct rounding of algebraic functions*. RAIRO Theoretical Informatics and Applications, vol. 41, 71-83, jan-mars 2007.

J-8. N. Brisebarre, J.-M. Muller, A. Tisserand et S. Torres. *Hardware Operators for Function Evaluation Using Sparse-Coefficient Polynomials*. Electronic Letters, vol. 42, Issue 25, 1441-1442, Dec. 2006.

J-9. N. Brisebarre et J.-M. Muller. *Correctly rounded multiplication by arbitrary precision constants*. IEEE Transactions on Computers, vol. **57**, n. 2, 165-174, 2008.

J-10. N. Brisebarre, J.-L. Beuchat, J. Detrey, E. Okamoto, M. Shirase et T. Takagi. *Algorithms and Arithmetic Operators for Computing the* η_T *Pairing in Characteristic Three*. IEEE Transactions on Computers, vol. **57**, n. 11, 1454-1468, 2008.

J-11. D. Wang, J.-M. Muller, N. Brisebarre et M. Ercegovac. (M, p, k)*-Friendly Points : a Table-Based Method to Evaluate Trigonometric Functions*. IEEE Transactions on Circuits and Systems II : express briefs, vol. **31**, n. 9, 711-715, Sep. 2014.

J-12. N. Brisebarre, C. Lauter, M. Mezzarobba et J.-M. Muller. *Comparison between binary64 and decimal64 floating-point numbers*. IEEE Transactions on Computers, vol. **65**, n. 7, 2032-2044, 2016.

J-13. B. Allombert, N. Brisebarre et A. Lasjaunias. *On a two-valued sequence and related continued fractions in power series fields*. The Ramanujan Journal, à paraître.

J-14. N. Brisebarre, G. Hanrot et O. Robert. *Exponential sums and and correctly-rounded functions*. IEEE Transactions on Computers, à paraître.

Articles dans des actes de conférences internationales

C-1. N. Brisebarre, J.-M. Muller et A. Tisserand. *Sparse-coefficient polynomial approximations for hardware implementations*. In Proc. 38th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, California, U.S.A., pages 532-535, Nov. 2004.

C-2. N. Brisebarre et J.-M. Muller. *Functions approximable by E-fractions*, (invited talk). In Proc. 38th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, California, U.S.A., pages 1341-1344, Nov. 2004.

C-3. N. Brisebarre et J.-M. Muller. *Correctly rounded multiplication by arbitrary precision constants*. 17th IEEE Symposium on Computer Arithmetic (ARITH-17), Cape Cod (MA, USA), pages 13-20, June 2005.

C-4. N. Brisebarre et S. Chevillard. *Efficient polynomial L^∞-approximations*. 18th IEEE Symposium on Computer Arithmetic (ARITH-18), Montpellier (France), pages 169-176, Juin 2007.

C-5. N. Brisebarre et G. Hanrot. *Floating-Point L^2-Approximations*. 18th IEEE Symposium on Computer Arithmetic (ARITH-18), Montpellier (France), pages 177-186, Juin 2007.

C-6. J.-L. Beuchat, N. Brisebarre, E. Okamoto, M. Shirase et T. Takagi. *A Coprocessor for the Final Exponentiation of the η_T Pairing in Characteristic Three*. In C. Carlet and B. Sunar, editors, Proceedings of Waifi 2007, n. 4547 of Lecture Notes in Computer Science, pages 25-39, 2007. Springer. Cryptology ePrint Archive, Report 2007/045, 2007.

C-7. J.-L. Beuchat, N. Brisebarre, J. Detrey et E. Okamoto. *Arithmetic Operators for Pairing-Based Cryptography*. In P. Paillier and I. Verbauwhede, editors, Proceedings of CHES 2007, n. 4727 of Lecture Notes in Computer Science, pages 239-255, 2007. Springer. **Note : Best paper award.** Cryptology ePrint Archive, Report 2007/091, 2007.

C-8. N. Brisebarre, S. Chevillard, M. Ercegovac, J.-M. Muller et S. Torres. *An Efficient Method for Evaluating Polynomial and Rational Function Approximations*. 19th IEEE Conference on Application-Specific Systems, Architectures and Processors (ASAP 2008). Leuven (Belgique), pages 233-238, July 2008.

C-9. N. Brisebarre, F. de Dinechin et J.-M. Muller. *Integer and Floating-Point Constant Multipliers for FPGAs*. 19th IEEE Conference on Application-Specific Systems, Architectures and Processors (ASAP 2008), pages 239-244. Leuven (Belgique), July 2008.

C-10. J.-L. Beuchat, N. Brisebarre, J. Detrey, E. Okamoto et F Rodríguez-Henríquez. *A Comparison between Hardware Accelerators for the Modified Tate Pairing over $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$*. Second International Conference on Pairing-based Cryptography (Pairing'08), Lecture Notes in Computer Science, pages 297-315. Egham, Royaume-Uni, Septembre 2008. Springer-Verlag.

C-11 N. Brisebarre, M. Ercegovac, N. Louvet, É. Martin-Dorel, J.-M. Muller et A. Panhaleux. *Implementing Decimal Floating-Point Arithmetic through Binary : some Suggestions*. 21st IEEE Conference on Application-Specific Systems, Architectures and Processors (ASAP 2010), pages 317–320. Rennes (France), July 2010.

C-12. N. Brisebarre et M. Joldeş. *Chebyshev Interpolation Polynomial-based Tools for Rigorous Computing*. 35th International Symposium on Symbolic and Algebraic Computation (ISSAC 2010), pages 147-154. Munich, Germany, July 2010.

C-13. N. Brisebarre, M. Joldeş, P. Kornerup, É. Martin-Dorel et J.-M. Muller. *Augmented precision square roots, 2-D norms, and discussion on correctly rounding* $\sqrt{x^2+y^2}$. 20th IEEE Symposium on Computer Arithmetic (ARITH-20), Tübingen (Germany), pages 23-30, July 2011.

C-14. N. Brisebarre, M. Joldeş, É. Martin-Dorel, M. Mayero, J.-M. Muller, Ioana Paşca, L. Rideau et L. Théry. *Rigorous Polynomial Approximation using Taylor Models in Coq*, pages 85-99. 4th NASA Formal Methods Symposium. Norfolk, Virginia, USA, April 2012.

C-15. N. Brisebarre, M. Ercegovac et J.-M. Muller. (M, p, k)*-friendly points : a table-based method for trigonometric function evaluation*. 23rd IEEE Conference on Application-Specific Systems, Architectures and Processors (ASAP 2012), pages 46-52. Delft (The Netherlands), July 2012.

C-16. N. Brisebarre, C. Lauter, M. Mezzarobba et J.-M. Muller. *Comparison between binary64 and decimal64 floating-point numbers*. 21st IEEE Symposium on Computer Arithmetic (ARITH-21), pages 145-152. April 2013.

Articles dans des actes de conférences nationales

NC-1. N. Brisebarre, S.-I. Filip et G. Hanrot. *De nouveaux résultats sur la synthèse de filtres RIF*. In Proc. 25ème colloque du Groupement de Recherche en Traitement du Signal et des Images (GRETSI), Lyon, Nov. 2015.

Articles soumis, rapports de recherche, travaux en cours

S-1. N. Brisebarre, *An algorithm for finding entire solutions of systems of difference equations*. Rapport de recherche LIP RR2003-53.

S-2. T. Györfi, O. Creţ, G. Hanrot et N. Brisebarre. *High-Throughput Hardware Architecture for the SWIFFT / SWIFFTX Hash Functions*, Cryptology ePrint Archive, Report 2012/343.

S-3. N. Brisebarre, S.-I. Filip et G. Hanrot. *A Lattice Basis Reduction Approach for the Design of Finite Wordlength FIR Filters*, soumis, https://hal.archives-ouvertes.fr/hal-01308801.

S-4. S.-I. Filip, M. Istoan, F. de Dinechin et N. Brisebarre. *Automatic Generation of Hardware FIR Filters From a Frequency Domain Specification*, https://hal.archives-ouvertes.fr/hal-01308377.

S-5. F. Bréhard, N. Brisebarre et M. Joldeş. *Validated and numerically efficient Chebyshev spectral methods for linear ordinary differential equations*, https://hal.archives-ouvertes.fr/hal-01526272.

S-6. N. Brisebarre, S. Chevillard et S. Torres, *Computing machine-number polynomial* L^∞*-approximations*, en préparation.

S-7. N. Brisebarre et G. Hanrot. *New results on the table maker's dilemma for transcendental functions*, en préparation.

Autres

A-1. N. Brisebarre. *Une étude de deux problèmes diophantiens*. Thèse, Université Bordeaux I, 1998.

A-2. Rédaction de la Leçon de Mathématiques d'Aujourd'hui "Fonctions modulaires et transcendance" donnée par Michel Waldschmidt le 30 novembre 1996 à Bordeaux. Leçons de Mathématiques d'aujourd'hui, Vol. 2, Le Sel et le Fer N. 12 (2003), leçon 5, 167-196, Éditions Cassini.

Annexe C

Encadrement d'étudiants

Thèse de Sylvain Chevillard

J'ai co-encadré avec J.-M. Muller la thèse de Sylvain Chevillard [1] (allocation couplée) débutée en septembre 2006 et soutenue en juillet 2009 [32]. Outre le travail en commun sur l'approximation polynomiale décrit dans ce qui précède, Sylvain a élaboré, d'un point de vue algorithmique et logiciel, une implantation de la fonction erf en arithmétique multi-précision qui est la plus performante à l'heure actuelle. Il a aussi conçu, notamment avec M. Joldeş, des algorithmes et des implantations d'une norme sup rapide et certifiée et a co-développé l'outil `Sollya`, qui est une boîte à outils pour le développeur de fonctions en logiciel et en matériel. Il a soutenu sa thèse le 6 juillet 2009 devant un jury composé de J.-P. Allouche (CNRS), B. Beckermann (Univ. Lille), B. Salvy (INRIA Rocquencourt), J.-M. Muller, P. Tang (INTEL) et moi-même. En relation avec sa thèse, Sylvain a publié un article dans deux journaux internationaux et cinq dans des conférences avec comité de lecture, dont deux dans ARITH, la conférence de référence d'arithmétique des ordinateurs. Après un post-doctorat d'une année au LORIA (Nancy), il a été recruté en 2010 chargé de recherche à l'INRIA, où il travaille au sein du projet APICS, au centre de recherche INRIA Sophia-Antipolis Méditerranée .

Thèse de Mioara Joldeş

J'ai co-encadré avec J.-M. Muller la thèse de Mioara Joldeş [2] (financée par une bourse de la région Rhône-Alpes) débutée en septembre 2008 et soutenue en septembre 2011 [98]. Mioara s'est intéressée au calcul d'erreurs certifiées d'approximations polynomiales, à la généralisation des « modèles de Taylor » (utilisés en arithmétique d'intervalles pour l'optimisation globale ou la résolution certifiée d'équations différentielles) à des « modèles de Tchebychev ». Elle a également étudié la génération d'approximations polynomiales adaptées à du calcul par matériel. Elle a soutenu sa thèse le 26 septembre 2011 devant un jury composé de F. Benhamou (Univ. Nantes), D. Henrion (CNRS LAAS), J.-M. Muller, W Tucker (Univ. Uppsala, Suède) et moi-même. En relation avec sa thèse, elle a publié ses travaux dans les actes des conférences ARITH'2009, FPL'2010, ICMS'2010, ISSAC'2010, ASAP'2010, ARITH'2011, NFM'2012 et a eu un article paru dans Theoretical Computer Science. De septembre 2011 à janvier 2013, elle a effectué un postdoctorat dans l'équipe de Warwick Tucker à l'université d'Uppsala, puis elle a intégré le CNRS en tant que CR2. Elle est en poste au LAAS (Toulouse), au sein de l'équipe MAC.

Thèse de Serge Torres

J'ai co-encadré avec J.-M. Muller la thèse de Serge Torres [3] (IR à l'ENS Lyon) [203]. Il l'a commencée en 2010 et l'a soutenue en septembre 2016 (Serge travaillait à temps partiel sur son doctorat, en complément de son activité d'ingénieur au LIP). Serge a travaillé sur l'évaluation sûre et efficace en machine et l'optimisation d'algorithmes de recherche de « pires cas » pour l'arrondi de fonctions (dans le cadre du projet ANR TaMaDi). Il a soutenu sa thèse le 22 septembre 2016 devant un jury composé de V. Berthé (CNRS LIAFA), Ph. Langlois (Univ. Perpignan), D. Michelucci (Univ. Bourgogne), J.-M. Muller, J. Villalba

1. http://www-sop.inria.fr/members/Sylvain.Chevillard/
2. http://homepages.laas.fr/mmjoldes/
3. http://perso.ens-lyon.fr/serge.torres/

(Univ. Malaga, Espagne) et moi-même. Son travail a fait l'objet d'un article dans le journal *Electronic Letters*, d'une publication dans les actes de la conférence ASAP'2008 et d'un article (sur l'arrondi de Ziv) publié à ACM TOMS. Il est également l'un des auteurs du *Handbook of Floating-Point Arithmetic* **B-1**. Nous rédigeons actuellement **S-6** avec S. Chevillard. Serge a repris son travail d'ingénieur de recherche au LIP à temps plein et devrait partir à la retraite en décembre prochain.

Thèse de Silviu-Ioan Filip

J'ai co-encadré avec G. Hanrot la thèse de Silviu-Ioan Filip [4] (contrat doctoral financé par l'ÉNS Lyon), commencée en septembre 2013 et soutenue en décembre 2016 [66]. Silviu a travaillé sur des questions d'approximation polynomiale et rationnelle et à leurs applications à la synthèse de filtres RIF et RII efficaces en machine. Il a développé depuis son stage de M2 une implantation robuste de l'algorithme de Parks-McClellan, qui est au cœur de la synthèse de filtres RIF et a avancé substantiellement sur la question de la quantification des coefficients de la réponse fréquentielle de ces filtres. Il a soutenu sa thèse le 7 décembre dernier devant un jury composé de B. Beckermann (Univ. Lille), J.-C. Belfiore (Huawei et Télécom Paris), G. Hanrot, J. Leblond (INRIA Sophia Antipolis), O. Sentieys (INRIA Rennes Bretagne Atlantique), L.N. Trefethen (Univ. Oxford) et moi-même. Il a publié un article de journal sur le premier point [67]. Silviu, G. Hanrot et moi-même avons soumis un article de journal sur la seconde question **S-3**. Enfin, en collaboration avec Silviu, F. de Dinechin et M. Istoan (INSA Lyon), nous avons développé un outil intégré en C++, pour la synthèse de filtres RIF sur circuits reconfigurables FPGAs. C'est la réalisation de la chaîne complète d'outils que nous avions initialement imaginée : partant des spécifications du filtre, l'outil renvoie du code VHDL optimisé d'évaluation du filtre. Ce travail est décrit dans **S-4**. Depuis le 1er septembre 2016, Silviu est post-doctorant au sein du Numerical Analysis Group, dirigé par L. N. Trefethen, de l'université d'Oxford (Royaume-Uni).

Thèse de Florent Bréhard

Je co-encadre avec Mioara Joldeş (LAAS, Toulouse) and Damien Pous (LIP) la thèse de Florent Bréhard (contrat doctoral spécifique normalien) sur le sujet *Outils pour un calcul certifié. Applications aux systèmes dynamiques et à la théorie du contrôle*. Florent a débuté son doctorat le 1er septembre 2016 en poursuivant un travail démarré au cours de sa 4ème année d'étude à l'ÉNS Paris sur la validation de solutions approchées d'équations différentielles linéaires à coefficients polynomiaux. Florent, M. Joldeş et moi sommes en train de terminer l'article **S-5**.

Post-doctorat de Marc Mezzarobba

Marc Mezzarobba [5] a effectué un post-doctorat financé par l'INRIA de décembre 2011 à mars 2013. Marc est un spécialiste de calcul symbolique et maîtrise de nombreux aspects du calcul numérique. Il a soutenu une excellente thèse (sous la direction de B. Salvy, INRIA Rocquencourt) en novembre 2011 et est venu dans notre équipe notamment pour développer des algorithmes et des programmes qui nous aident à mettre au point des outils de générations entièrement automatisée de code quasi-optimal d'évaluation de fonction.

Pendant son séjour chez nous, il a publié un article dans les actes de la conférence CASC'2012 et deux articles à la conférence ARITH'2012. Dans la foulée de son post-doc chez nous, il a été recruté en tant que CR2 CNRS au LIP6, au sein de l'équipe Péquan, qu'il a intégrée après quelques mois de post-doc supplémentaires au RISC (Linz, Autriche).

4. http://perso.ens-lyon.fr/silviuioan.filip/
5. http://www.marc.mezzarobba.net/

Bibliographie

[1] D. Aggarwal, D. Dadush, and N. Stephens-Davidowitz. Solving the Closest Vector Problem in 2^n Time – The Discrete Gaussian Strikes Again ! In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 563–582. IEEE, 2015.

[2] M. Ajtai. The Shortest Vector Problem in L_2 is NP-hard for Randomized Reductions (Extended Abstract). In *Proceedings of the 30th ACM symposium on Theory of computing (STOC)*, pages 10–19, 1998.

[3] American National Standards Institute and Institute of Electrical and Electronic Engineers. *IEEE Standard for Binary Floating-Point Arithmetic*. ANSI/IEEE Standard 754–1985, 1985.

[4] American National Standards Institute and Institute of Electrical and Electronic Engineers. *IEEE Standard for Radix Independent Floating-Point Arithmetic*. ANSI/IEEE Standard 854–1987, 1987.

[5] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' guide*, volume 9. SIAM, 1999.

[6] A. Antoniou. *Digital Signal Processing : Signals, Systems, and Filters*. McGraw-Hill Education, 2005.

[7] L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1) :1–13, 1986.

[8] P. S. L. M. Barreto, S. D. Galbraith, C. Ó hÉigeartaigh, and M. Scott. Efficient pairing computation on supersingular Abelian varieties. In *Designs, Codes and Cryptography*, volume 42(3), pages 239–271. Springer Netherlands, Mar. 2007.

[9] G. Baszenski and M. Tasche. Fast polynomial multiplication and convolutions related to the discrete cosine transform. *Linear Algebra Appl.*, 252 :1–25, 1997.

[10] M. Bellanger. *Traitement numérique du signal*. Dunod, 2012.

[11] C. M. Bender and S. A. Orszag. *Advanced mathematical methods for scientists and engineers. I*. Springer-Verlag, New York, 1999. Asymptotic methods and perturbation theory, Reprint of the 1978 original.

[12] C. Bendsten and O. Stauning. TADIFF, a Flexible C++ Package for Automatic Differentiation Using Taylor Series. Technical Report 1997-x5-94, Technical Univ. of Denmark, April 1997.

[13] A. Benoit, M. Joldeş, and M. Mezzarobba. Rigorous uniform approximation of D-finite functions using Chebyshev expansions. *Mathematics of Computation*, 86 :1303–1341, 2017.

[14] S. N. Bernstein. *Leçons sur les propriétés extrémales et la meilleure approximation des fonctions analytiques d'une variable réelle*. Gauthier-Villars, 1926.

[15] M. Berz and K. Makino. Rigorous global search using Taylor models. In *SNC '09 : Proceedings of the 2009 conference on Symbolic numeric computation*, pages 11–20, New York, NY, USA, 2009. ACM.

[16] M. Berz, K. Makino, and Y.-K. Kim. Long-term stability of the Tevatron by verified global optimization. *Nuclear Instruments and Methods in Physics Research Section A : Accelerators, Spectrometers, Detectors and Associated Equipment*, 558(1) :1 – 10, 2006. Proceedings of the 8th International Computational Accelerator Physics Conference - ICAP 2004.

[17] J.-P. Bézivin and F. Gramain. Solutions entières d'un système d'équations aux différences. *Ann. Inst. Fourier (Grenoble)*, 43(3) :791–814, 1993.

[18] J.-P. Bézivin and F. Gramain. Solutions entières d'un système d'équations aux différences. II. *Ann. Inst. Fourier (Grenoble)*, 46(2) :465–491, 1996.

[19] E. Bombieri and H. Iwaniec. On the order of $\zeta(\frac{1}{2} + it)$. *Ann. Scuola Norm. Sup. Pisa Cl. Sci. (4)*, 13(3) :449–472, 1986.

[20] E. Bombieri and H. Iwaniec. Some mean-value theorems for exponential sums. *Ann. Scuola Norm. Sup. Pisa Cl. Sci. (4)*, 13(3) :473–486, 1986.

[21] L. P. Bos, J.-P. Calvi, N. Levenberg, A. Sommariva, and M. Vianello. Geometric weakly admissible meshes, discrete least squares approximations and approximate Fekete points. *Mathematics of Computation*, 80(275) :1623–1638, 2011.

[22] L. P. Bos and N. Levenberg. On the calculation of approximate Fekete points : the univariate case. *Electronic Transactions on Numerical Analysis*, 30 :377–397, 2008.

[23] A. Bostan, B. Salvy, and E. Schost. Fast conversion algorithms for orthogonal polynomials. *Linear Algebra and its Applications*, 432(1) :249–258, January 2010.

[24] J. Bourgain. Decoupling, exponential sums and the Riemann zeta function. *J. Amer. Math. Soc.*, 30(1) :205–224, 2017.

[25] J. P. Boyd. *Chebyshev and Fourier spectral methods*. Dover Publications Inc., Mineola, NY, second edition, 2001. available from http://www-personal.umich.edu/~jpboyd/BOOK_Spectral2000.html.

[26] J.-P. Calvi and N. Levenberg. Uniform approximation by discrete least squares polynomials. *Journal of Approximation Theory*, 152(1) :82–100, May 2008.

[27] Y. Cao, K. Wang, W. Pei, Y. Liu, and Y. Zhang. Design of high-order extrapolated impulse response FIR filters with signed powers-of-two coefficients. *Circuits, Systems, and Signal Processing*, 30(5) :963–985, 2011.

[28] J. W. S. Cassels. *An introduction to the geometry of numbers*. Classics in Mathematics. Springer-Verlag, Berlin, 1997. Corrected reprint of the 1971 edition.

[29] A. Çivril and M. Magdon-Ismail. On selecting a maximum volume sub-matrix of a matrix and related problems. *Theoretical Computer Science*, 410(47-49) :4801–4811, Nov. 2009.

[30] C.-Y. Chen. Computing interval enclosures for definite integrals by application of triple adaptive strategies. *Computing*, pages 81–99, 2006.

[31] E. W. Cheney. *Introduction to approximation theory*. AMS Chelsea Publishing, Providence, RI, 1998. Reprint of the second (1982) edition.

[32] S. Chevillard. *Évaluation efficace de fonctions numériques – Outils et exemples*. PhD thesis, École normale supérieure de Lyon – Université de Lyon, 2009. In French.

[33] S. Chevillard, J. Harrison, M. Joldeş, and C. Lauter. Efficient and accurate computation of upper bounds of approximation errors. *Theoretical Computer Science*, 412(16) :1523–1543, 2011.

[34] S. Chevillard, M. Joldes, and C. Lauter. Certified and fast computation of supremum norms of approximation errors. In *19th IEEE Symposium on Computer Arithmetic*, pages 169–176, 2009.

[35] D. V. Chudnovsky and G. V. Chudnovsky. Hypergeometric and modular function identities, and new rational approximations to and continued fraction expansions of classical constants and functions. In *A tribute to Emil Grosswald : number theory and related analysis*, volume 143 of *Contemp. Math.*, pages 117–162. Amer. Math. Soc., Providence, RI, 1993.

[36] W. J. Cody. A proposed radix and word length independent standard for floating-point arithmetic. *ACM SIGNUM Newsletter*, 20 :37–51, Jan. 1985.

[37] W. J. Cody, J. T. Coonen, D. M. Gay, K. Hanson, D. Hough, W. Kahan, R. Karpinski, J. Palmer, F. N. Ris, and D. Stevenson. A proposed radix-and-word-length-independent standard for floating-point arithmetic. *IEEE MICRO*, 4(4) :86–100, Aug. 1984.

[38] H. Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1993.

[39] J. H. Conway and N. J. A. Sloane. *Sphere packings, lattices and groups*, volume 290 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, New York, third edition, 1999. With additional contributions by E. Bannai, R. E. Borcherds, J. Leech, S. P. Norton, A. M. Odlyzko, R. A. Parker, L. Queen and B. B. Venkov.

[40] M. Cornea, J. Harrison, and P. T. P. Tang. *Scientific Computing on Itanium-Based Systems*. Intel Press, 2002.

[41] E. Croot, R.-C. Li, and H. J. Zhu. The abc conjecture and correctly rounded reciprocal square roots. *Theoretical Computer Science*, 315(2-3) :405–417, May 2004.

[42] E. A. da Silva, L. Lovisolo, A. J. Dutra, and P. S. Diniz. FIR filter design based on successive approximation of vectors. *IEEE Transactions on Signal Processing*, 62(15) :3833–3848, 2014.

[43] D. Dadush and G. Kun. Lattice sparsification and the approximate closest vector problem. *Theory of Computing*, 12(1) :1–34, 2016.

[44] M. Daumas, C. Mazenc, X. Merrheim, and J.-M. Muller. Modular range reduction : A new algorithm for fast and accurate computation of the elementary functions. *Journal of Universal Computer Science*, 1(3) :162–175, Mar. 1995.

[45] M. Daumas and G. Melquiond. Certification of bounds on expressions involving rounded operators. *Transactions on Mathematical Software*, 37(1) :1–20, 2010.

[46] F. de Dinechin, A. V. Ershov, and N. Gast. Towards the post-ultimate libm. In *Proceedings of the 17th IEEE Symposium on Computer Arithmetic*, ARITH '05, pages 288–295, Washington, DC, USA, 2005. IEEE Computer Society.

[47] F. de Dinechin, C. Lauter, and G. Melquiond. Certifying the floating-point implementation of an elementary function using Gappa. *Transactions on Computers*, 60(2) :242–253, 2011.

[48] F. de Dinechin, C. Q. Lauter, and J.-M. Muller. Fast and correctly rounded logarithms in double-precision. *Theoretical Informatics and Applications*, 41 :85–102, 2007.

[49] J.-P. Demailly. *Analyse numérique et équations différentielles- 4ème édition*. EDP sciences, 2016.

[50] J. Descloux. *Contribution au calcul des approximations de Tschebischeff*. PhD thesis, École polytechnique fédérale de Lausanne (EPFL), 1960.

[51] J. Descloux. Dégénérescence dans les approximations de Tschebyscheff linéaires et discrètes. *Numerische Mathematik*, 3(1) :180–187, 1961.

[52] P. Di Lizia. *Robust Space Trajectory and Space System Design using Differential Algebra*. PhD thesis, Politecnico di Milano, Milano, Italy, 2008.

[53] I. Dinur, G. Kindler, R. Raz, and S. Safra. Approximating CVP to Within Almost-Polynomial Factors is NP-Hard. *Combinatorica*, 23(2) :205–243, 2003.

[54] C. K. Dubey and T. Holenstein. Approximating the closest vector problem using an approximate shortest vector oracle. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 14th International Workshop, APPROX 2011, and 15th International Workshop, RANDOM 2011, Princeton, NJ, USA, August 17-19, 2011. Proceedings*, pages 184–193, 2011.

[55] A. K. Dubitskas. Approximation of $\pi/\sqrt{3}$ by rational fractions. *Vestnik Moskov. Univ. Ser. I Mat. Mekh.*, (6) :73–76, 1987.

[56] A. K. Dubitskas. On improving the approximation of $\pi/\sqrt{3}$ by rational fractions. *Vestnik Moskov. Univ. Ser. I Mat. Mekh.*, (6) :76–77 (1994), 1993.

[57] C. B. Dunham. Feasibility of "perfect" function evaluation. *SIGNUM Newsletter*, 25(4) :25–26, Oct. 1990.

[58] D. Elliott, D. F. Paget, G. M. Phillips, and P. J. Taylor. Error of truncated Chebyshev series and other near minimax polynomial approximations. *J. Approx. Theory*, 50(1) :49–57, 1987.

[59] C. Epstein, W. Miranker, and T. Rivlin. Ultra-arithmetic I : function data types. *Mathematics and Computers in Simulation*, 24(1) :1–18, 1982.

[60] C. Epstein, W. Miranker, and T. Rivlin. Ultra-arithmetic II : intervals of polynomials. *Mathematics and Computers in Simulation*, 24(1) :19–29, 1982.

[61] M. D. Ercegovac. *A general method for evaluation of functions and computation in a digital computer*. PhD thesis, Dept. of Computer Science, University of Illinois, Urbana-Champaign, IL, 1975.

[62] M. D. Ercegovac. A general hardware-oriented method for evaluation of functions and computations in a digital computer. *IEEE Transactions on Computers*, C-26(7) :667–680, 1977.

[63] G. Evangelista. Design of optimum high-order finite-wordlength digital FIR filters with linear phase. *Signal Processing*, 82(2) :187–194, 2002.

[64] R. T. Farouki. The Bernstein polynomial basis : a centennial retrospective. *Comput. Aided Geom. Design*, 29(6) :379–419, 2012. available from http://mae.engr.ucdavis.edu/~farouki/bernstein.pdf.

[65] N. I. Fel'dman. An effective power sharpening of a theorem of Liouville. *Izv. Akad. Nauk SSSR Ser. Mat.*, 35 :973–990, 1971.

[66] S.-I. Filip. *Robust tools for weighted Chebyshev approximation and applications to digital filter design*. PhD thesis, École Normale Supérieure de Lyon, 2016.

[67] S.-I. Filip. A robust and scalable implementation of the Parks-McClellan algorithm for designing FIR filters. *ACM Transactions on Mathematical Software*, 43(1), Aug. 2016.

[68] U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Math. Comp.*, 44(170) :463–471, 1985.

[69] L. Fox and I. B. Parker. *Chebyshev polynomials in numerical analysis*. Oxford University Press, London-New York-Toronto, Ont., 1968.

[70] S. Gal and B. Bachelis. An accurate elementary mathematical library for the IEEE floating point standard. *ACM Transactions on Mathematical Software*, 17(1) :26–45, Mar. 1991.

[71] S. Galbraith. Pairings. In *Advances in elliptic curve cryptography*, volume 317 of *London Math. Soc. Lecture Note Ser.*, pages 183–213. Cambridge Univ. Press, Cambridge, 2005.

[72] W. Gautschi. *Numerical analysis*. Birkhäuser Boston Inc., Boston, MA, 1997. An introduction.

[73] P. Giorgi. On polynomial multiplication in Chebyshev basis. *IEEE Trans. Comput.*, 61(6) :780–789, 2012.

[74] O. Goldreich and S. Goldwasser. On the limits of non-approximability of lattice problems. In *Proceedings of 30th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–9. ACM, May 1998.

[75] O. Goldreich, D. Micciancio, S. Safra, and J. Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Inf. Process. Lett.*, 71(2) :55–61, 1999.

[76] D. Gottlieb and S. A. Orszag. *Numerical Analysis of Spectral Methods : Theory and Applications*, volume 26. Siam, 1977.

[77] S. W. Graham and G. Kolesnik. *Van der Corput's method of exponential sums*. Cambridge University Press, 1991.

[78] A. Griewank. A mathematical view of automatic differentiation. *Acta Numer.*, 12 :321–398, 2003.

[79] A. Griewank and A. Walther. *Evaluating derivatives*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2008. Principles and techniques of algorithmic differentiation.

[80] P. M. Gruber and C. G. Lekkerkerker. *Geometry of numbers*, volume 37 of *North-Holland Mathematical Library*. North-Holland Publishing Co., Amsterdam, second edition, 1987.

[81] O. Gustafsson and L. Wanhammar. Design of linear-phase FIR filters combining subexpression sharing with MILP. In *The 2002 45th Midwest Symposium on Circuits and Systems, 2002. MWSCAS-2002.*, volume 3, pages 9–12, Aug. 2002.

[82] T. C. Hales. A proof of the Kepler conjecture. *Ann. of Math. (2)*, 162(3) :1065–1185, 2005.

[83] T. C. Hales, J. Harrison, S. McLaughlin, T. Nipkow, S. Obua, and R. Zumkeller. A revision of the proof of the Kepler conjecture. *Discrete Comput. Geom.*, 44(1) :1–34, 2010.

[84] G. Hanrot, X. Pujol, and D. Stehlé. Algorithms for the shortest and closest lattice vector problems. In Y. M. Chee, Z. Guo, S. Ling, F. Shao, Y. Tang, H. Wang, and C. Xing, editors, *Coding and Cryptology - Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings*, volume 6639 of *Lecture Notes in Computer Science*, pages 159–190. Springer, 2011.

[85] G. Hanrot, X. Pujol, and D. Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 447–464, 2011.

[86] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, London, 1979.

[87] J. Harrison. Decimal transcendentals via binary. In *Proceedings of the 19th IEEE Symposium on Computer Arithmetic (ARITH-19)*. IEEE Computer Society Press, June 2009.

[88] J. Hass and R. Schlafly. Double bubbles minimize. *Ann. of Math. (2)*, 151(2) :459–515, 2000.

[89] M. Hata. Legendre type polynomials and irrationality measures. *J. Reine Angew. Math.*, 407 :99–125, 1990.

[90] C. Hermite. Extraits de lettres de M. Hermite à M. Jacobi sur différents objets de la théorie des nombres, deuxième lettre. *Journal für die reine und angewandte Mathematik*, 40 :279–290, 1850.

[91] O. Herrmann. Über die Berechnung der Fourierkoeffizienten der Funktion $j(\tau)$. *J. Reine Angew. Math.*, 274/275 :187–195, 1975. Collection of articles dedicated to Helmut Hasse on his seventy-fifth birthday, III.

[92] D. Hilbert. Mathematische Probleme. Vortrag, gehalten auf dem internationalen Mathematiker-Congress zu Paris 1900. *Nachr. Ges. Wiss. Göttingen, Math.-Phys. Kl.*, 1900 :253–297, 1900.

[93] A. Hurwitz. Ueber die angenäherte Darstellung der Irrationalzahlen durch rationale Brüche. *Math. Ann.*, 39(2) :279–284, 1891.

[94] M. N. Huxley. *Area, lattice points, and exponential sums*, volume 13. Clarendon Press, 1996.

[95] IEEE Computer Society. *IEEE Standard for Floating-Point Arithmetic*. IEEE Standard 754-2008, Aug. 2008. available at `http://ieeexplore.ieee.org/servlet/opac?punumber=4610933`.

[96] C. Iordache and D. W. Matula. On infinitely precise rounding for division, square root, reciprocal and square root reciprocal. In Koren and Kornerup, editors, *Proceedings of the 14th IEEE Symposium on Computer Arithmetic (Adelaide, Australia)*, pages 233–240. IEEE Computer Society Press, Los Alamitos, CA, Apr. 1999.

[97] T. Johnson and W. Tucker. On a computer-aided approach to the computation of Abelian integrals. *BIT*, 51(3) :653–667, 2011.

[98] M. Joldeş. *Rigorous Polynomial Approximations and Applications*. PhD thesis, École normale supérieure de Lyon – Université de Lyon, Lyon, France, 2011.

[99] A. Joux. *La Réduction de Réseaux en Cryptographie*. PhD thesis, École Normale Supérieure, 1993.

[100] W. Kahan. Why do we need a floating-point standard ? Technical report, Computer Science, UC Berkeley, 1981. Available at `http://www.cs.berkeley.edu/~wkahan/ieee754status/why-ieee.pdf`.

[101] W. Kahan. A logarithm too clever by half. Available at `http://http.cs.berkeley.edu/~wkahan/LOG10HAF.TXT`, 2004.

[102] R. Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the annual ACM symposium on theory of computing (STOC)*, pages 193–206, 1983.

[103] R. Kannan. Algorithmic geometry of numbers. *Annual Review of Computer Science*, 2 :231–267, 1987.

[104] R. Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3) :415–440, Aug. 1987.

[105] E. W. Kaucher and W. L. Miranker. *Self-validating numerics for function space problems : Computation with guarantees for differential and integral equations*, volume 9. Elsevier, 2014.

[106] R. B. Kearfott. *Rigorous global search : continuous problems*. Kluwer, Dordrecht, 1996.

[107] A. Y. Khinchin. *Continued Fractions*. Dover, New York, 1997.

[108] A. Khintchine. Einige Sätze über Kettenbrüche, mit Anwendungen auf die Theorie der diophantischen Approximationen. *Mathematische Annalen*, 92(1) :115–125, Mar. 1924.

[109] J. Knowles and E. Olcayto. Coefficient accuracy and digital filter response. *IEEE Transactions on Circuit Theory*, 15(1) :31–41, 1968.

[110] D. M. Kodek. Design of optimal finite wordlength FIR digital filters using integer programming techniques. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(3) :304–308, June 1980.

[111] D. M. Kodek. Design of optimal finite wordlength FIR digital filters. In *Proc. of the European Conference on Circuit Theory and Design, ECCTD '99.*, volume 1, pages 401–404, Aug. 1999.

[112] D. M. Kodek. Performance limit of finite wordlength FIR digital filters. *IEEE Transactions on Signal Processing*, 53(7) :2462–2469, July 2005.

[113] D. M. Kodek. LLL Algorithm and the Optimal Finite Wordlength FIR Design. *IEEE Transactions on Signal Processing*, 60(3) :1493–1498, Mar. 2012.

[114] D. M. Kodek and M. Krisper. Telescoping rounding for suboptimal finite wordlength FIR digital filter design. *Digital Signal Processing*, 15(6) :522–535, 2005.

[115] D. M. Kodek and K. Steiglitz. Comparison of Optimal and Local Search Methods for Designing Finite Wordlength FIR Digital Filters. *IEEE Transactions on Circuits and Systems*, 28(1) :28–32, 1981.

[116] E. Krätzel. *Lattice points*, volume 33 of *Mathematics and its Applications*. Springer, 1989.

[117] J. C. Lagarias. *The Kepler conjecture : the Hales-Ferguson proof*. Springer Science+Business Media, LLC, New York, NY, 2011.

[118] J. C. Lagarias, H. W. Lenstra, and C. P. Schnorr. Korkin-Zolotarev bases and successive minima of a lattice and its reciprocal lattice. *Combinatorica*, 10(4) :333–348, 1990.

[119] J.-L. Lagrange. Recherches d'arithmétique. In J.-A. Serret, editor, *Œuvres*, volume 3, pages 695–795. Gauthier Villars, Paris, 1867.

[120] T. Lang and J.-M. Muller. Bound on run of zeros and ones for algebraic functions. In N. Burgess and L. Ciminiera, editors, *Proceedings of the 15th IEEE Symposium on Computer Arithmetic (ARITH-16)*, pages 13–20, June 2001.

[121] A. Lasjaunias. A survey of Diophantine approximation in fields of power series. *Monatsh. Math.*, 130(3) :211–229, 2000.

[122] A. Lasjaunias. Diophantine approximation and continued fractions in power series fields. In *Analytic number theory*, pages 297–305. Cambridge Univ. Press, Cambridge, 2009.

[123] C. Q. Lauter. *Arrondi Correct de Fonctions Mathématiques*. PhD thesis, École Normale Supérieure de Lyon, Lyon, France, Oct. 2008. In French.

[124] C. Q. Lauter and V. Lefèvre. An efficient rounding boundary test for $\mathrm{pow}(x, y)$ in double precision. *IEEE Transactions on Computers*, 58(2) :197–207, Feb. 2009.

[125] V. Lefèvre. *Moyens Arithmétiques Pour un Calcul Fiable*. PhD thesis, École Normale Supérieure de Lyon, Lyon, France, 2000.

[126] V. Lefèvre and J.-M. Muller. Worst cases for correct rounding of the elementary functions in double precision. In N. Burgess and L. Ciminiera, editors, *Proceedings of the 15th IEEE Symposium on Computer Arithmetic (ARITH-16)*, pages 111–118, Vail, CO, June 2001.

[127] V. Lefèvre, J.-M. Muller, and A. Tisserand. Towards correctly rounded transcendentals. In *Proceedings of the 13th IEEE Symposium on Computer Arithmetic*, pages 132–137. IEEE Computer Society Press, Los Alamitos, CA, 1997.

[128] A.-M. Legendre. *Essai sur la théorie des nombres*. Cambridge Library Collection. Cambridge University Press, Cambridge, 2009. Reprint of the second (1808) edition.

[129] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4) :515–534, 1982.

[130] Y. Lim. Design of discrete-coefficient-value linear phase FIR filters with optimum normalized peak ripple magnitude. *IEEE Transactions on Circuits and Systems*,, 37(12) :1480–1486, Dec. 1990.

[131] Y. C. Lim and A. Constantinides. New integer programming scheme for nonrecursive digital filter design. *Electronics Letters*, 15(25) :812–813, Dec. 1979.

[132] Y. C. Lim and S. Parker. Discrete Coefficient Fir Digital Filter Design Based Upon an LMS Criteria. *IEEE Transactions on Circuits and Systems*, 30(10) :723–739, Oct. 1983.

[133] Y. C. Lim and S. Parker. FIR filter design over a discrete powers-of-two coefficient space. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 31(3) :583–591, June 1983.

[134] Y. C. Lim, S. Parker, and A. Constantinides. Finite word length FIR filter design using integer programming over a discrete coefficient space. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 30(4) :661–664, Aug. 1982.

[135] Y. C. Lim, R. Yang, D. Li, and J. Song. Signed power-of-two term allocation scheme for the design of digital filters. *IEEE Transactions on Circuits and Systems II : Analog and Digital Signal Processing*, 46(5) :577–584, May 1999.

[136] J. Liouville. Nouvelle démonstration d'un théorème sur les irrationnelles algébriques. *C. R. Acad. Sci. Paris*, 18 :910–911, 1844.

[137] J. Liouville. Remarques relatives à des classes très-étendues de quantités dont la valeur n'est ni algébrique ni même réductible à des irrationnelles algébriques. *C. R. Acad. Sci. Paris*, 18 :883–885, 1844.

[138] J. Liouville. Sur des classes très étendues de quantités dont la valeur n'est ni algébrique ni même réductible à des irrationnelles algébriques. *J. Math. Pures Appl.*, 16 :133–142, 1851.

[139] L. Lipshitz. *D*-finite power series. *Journal of algebra*, 122(2) :353–373, 1989.

[140] L. Lovász. *An algorithmic theory of numbers, graphs and convexity*, volume 50 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1986.

[141] K. Mahler. On the coefficients of transformation polynomials for the modular function. *Bull. Austral. Math. Soc.*, 10 :197–218, 1974.

[142] K. Makino and M. Berz. Taylor models and other validated functional inclusion methods. *International Journal of Pure and Applied Mathematics*, 4(4) :379–456, 2003.

[143] P. Markstein. *IA-64 and Elementary Functions : Speed and Precision*. Hewlett-Packard Professional Books. Prentice-Hall, Englewood Cliffs, NJ, 2000.

[144] P. W. Markstein. The New IEEE-754 Standard for Floating-Point Arithmetic. In *Numerical Validation in Current Hardware Architectures*, number 08021 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2008.

[145] J. Martinet. *Perfect lattices in Euclidean spaces*, volume 327 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 2003.

[146] J. C. Mason and D. C. Handscomb. *Chebyshev polynomials*. CRC Press, 2002.

[147] G. Melquiond. *De l'arithmétique d'intervalles à la certification de programmes*. PhD thesis, École Normale Supérieure de Lyon, Lyon (France), 2006.

[148] F. Messine. *Méthodes d'optimisation globale basées sur l'analyse d'intervalle pour la résolution de problèmes avec contraintes*. PhD thesis, INP de Toulouse, 1997.

[149] D. Micciancio. The shortest vector problem is NP-hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6) :2008–2035, Mar. 2001. Preliminary version in FOCS 1998.

[150] D. Micciancio and S. Goldwasser. *Complexity of lattice problems : a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, MA, 2002.

[151] H. L. Montgomery. *Ten lectures on the interface between analytic number theory and harmonic analysis*. Number 84 in Conference board of the Mathematical Sciences. American Math. Soc., 1994.

[152] R. E. Moore. *Interval Analysis*. Prentice-Hall, 1966.

[153] R. E. Moore and F. Bierbaum. *Methods and Applications of Interval Analysis*, volume 2. SIAM, 1979.

[154] I. Morel, D. Stehlé, and G. Villard. Using Householder inside LLL. In *Proceeding of ISSAC'09*, pages 271–278, Seoul (Korea), July 2009.

[155] J.-M. Muller. *Elementary Functions, Algorithms and Implementation*. Birkhäuser, Boston, 3rd edition, 2016.

[156] J.-M. Muller, N. Brisebarre, F. de Dinechin, C.-P. Jeannerod, V. Lefèvre, G. Melquiond, N. Revol, D. Stehlé, and S. Torres. *Handbook of Floating-Point Arithmetic*. Birkhäuser, 2010.

[157] P. S. V. Nataraj and K. Kotecha. Global optimization with higher order inclusion function forms part 1 : A combined taylor-bernstein form. *Reliable Computing*, 10(1) :27–44, 2004.

[158] Y. V. Nesterenko and M. Waldschmidt. On the approximation of the values of exponential function and logarithm by algebraic numbers (in Russian). *Mat. Zapiski*, 2 :23–42, 1996. Available in English at `http://www.math.jussieu.fr/~miw/articles/ps/Nesterenko.ps`.

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

[160] K. C. Ng. Argument reduction for huge arguments : Good to the last bit. Technical report, SunPro, 1992. Can be obtained by sending an e-mail to the author : `kwok.ng@eng.sun.com`.

[161] P. Nguyen and D. Stehlé. Floating-point LLL revisited. In *Proceedings of Eurocrypt 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 215–233. Springer-Verlag, 2005.

[162] P. Q. Nguyen. *La Géométrie des Nombres en Cryptologie*. PhD thesis, École Normale Supérieure, 1999.

[163] P. Q. Nguyen and D. Stehlé. An LLL algorithm with quadratic complexity. *SIAM J. Comput.*, 39(3) :874–903, 2009.

[164] P. Q. Nguyen and B. Vallée, editors. *The LLL Algorithm - Survey and Applications*. Information Security and Cryptography. Springer, 2010.

[165] J. Nielsen. Design of linear-phase direct-form FIR digital filters with quantized coefficients using error spectrum shaping. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(7) :1020–1026, July 1989.

[166] T. H. I. U. of Crystallography. *Space-group symmetry*, volume A of *International Tables for Crystallography*. Springer Netherlands, 2002.

[167] S. Olver and A. Townsend. A fast and well-conditioned spectral method. *SIAM Review*, 55(3) :462–489, 2013.

[168] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing*. Prentice-Hall Signal Processing Series. Prentice Hall, 2010.

[169] T. W. Parks and J. H. McClellan. Chebyshev Approximation for Nonrecursive Digital Filters with Linear Phase. *IEEE Transactions on Circuit Theory*, 19(2) :189–194, March 1972.

[170] M. Payne and R. Hanek. Radian reduction for trigonometric functions. *SIGNUM Newsletter*, 18 :19–24, 1983.

[171] M. Petkovšek, H. S. Wilf, and D. Zeilberger. $A = B$. A K Peters, Ltd., Wellesley, MA, 1996. With a foreword by Donald E. Knuth, With a separately available computer disk.

[172] A. Pinkus. Weierstrass and approximation theory. *J. Approx. Theory*, 107(1) :1–66, 2000. available from `http://www.math.technion.ac.il/hat/fpapers/wap.pdf`.

[173] R. B. Platte and L. N. Trefethen. Chebfun : a new kind of numerical computing. In *Progress in industrial mathematics at ECMI 2008*, volume 15 of *Math. Ind.*, pages 69–87. Springer, Heidelberg, 2010.

[174] G. Plonka and M. Tasche. Fast and numerically stable algorithms for discrete cosine transforms. *Linear Algebra Appl.*, 394 :309–345, 2005.

[175] M. J. D. Powell. *Approximation theory and methods*. Cambridge University Press, 1981.

[176] P. Prandoni and M. Vetterli. *Signal Processing for Communications*. Taylor & Francis, 2008.

[177] L. B. Rall. The arithmetic of differentiation. *Mathematics Magazine*, 59(5) :275–282, 1986.

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

[179] E. Remez. Sur un procédé convergent d'approximations successives pour déterminer les polynômes d'approximation (in french). *C.R. Académie des Sciences, Paris*, 198 :2063–2065, 1934.

[180] D. Ridout. Rational approximations to algebraic numbers. *Mathematika*, 4 :125–131, 1957.

[181] T. J. Rivlin. *The Chebyshev Polynomials*. Wiley, 1974.

[182] O. Robert. On van der Corput's k-th derivative test for exponential sums. *Indag. Math. (N.S.)*, 27(2) :559–589, 2016.

[183] K. F. Roth. Rational approximations to algebraic numbers. *Mathematika*, 2 :1–20, 1955.

[184] F. Rouillier and P. Zimmermann. Efficient isolation of polynomial's real roots. *Journal of Computational and Applied Mathematics*, 162(1) :33–50, 2004.

[185] E. A. Rukhadze. A lower bound for the approximation of $\ln 2$ by rational numbers. *Vestnik Moskov. Univ. Ser. I Mat. Mekh.*, (6) :25–29, 97, 1987.

[186] M. Schatzman. *Numerical Analysis, A Mathematical Introduction*. Oxford University Press, 2002.

[187] C. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53 :201–224, 1987.

[188] A. Schrijver. *Combinatorial optimization. Polyhedra and efficiency. Vol. A*. Algorithms and Combinatorics, 24. Springer-Verlag, Berlin, 2003.

[189] I. A. Semaev. A 3-dimensional lattice reduction algorithm. In *Proceedings of the 2001 Cryptography and Lattices Conference (CaLC'01)*, volume 2146 of *Lecture Notes in Computer Science*, pages 181–193. Springer, 2001.

[190] J.-P. Serre. *A Course in Arithmetic*. Springer-Verlag, New York-Heidelberg, 1973. Translated from the French, Graduate Texts in Mathematics, No. 7.

[191] D. Shi and Y. J. Yu. Design of Linear Phase FIR Filters With High Probability of Achieving Minimum Number of Adders. *IEEE Transactions on Circuits and Systems I : Regular Papers*, 58(1) :126–136, Jan. 2011.

[192] J. Skaf and S. P. Boyd. Filter Design With Low Complexity Coefficients. *IEEE Transactions on Signal Processing*, 56(7) :3162–3169, July 2008.

[193] A. Sommariva and M. Vianello. Computing approximate Fekete points by QR factorizations of Vandermonde matrices. *Computers & Mathematics with Applications*, 57(8) :1324–1336, Apr. 2009.

[194] A. Sommariva and M. Vianello. Approximate Fekete points for weighted polynomial interpolation. *Electronic Transactions on Numerical Analysis*, 37 :1–22, 2010.

[195] D. Stehlé. On the Randomness of Bits Generated by Sufficiently Smooth Functions. In F. Hess, S. Pauli, and M. E. Pohst, editors, *Algorithmic Number Theory, 7th International Symposium, ANTS-VII, Berlin, Germany, July 23-28, 2006, Proceedings*, volume 4076 of *Lecture Notes in Computer Science*, pages 257–274. Springer, 2006.

[196] D. Stehlé, V. Lefèvre, and P. Zimmermann. Worst Cases and Lattice Reduction. In J.-C. Bajard and M. J. Schulte, editors, *Proceedings of the 16th IEEE Symposium on Computer Arithmetic (ARITH-16)*, pages 142–147. IEEE Computer Society Press, Los Alamitos, CA, June 2003.

[197] D. Stehlé, V. Lefèvre, and P. Zimmermann. Searching Worst Cases of a One-Variable Function Using Lattice Reduction. *IEEE Transactions on Computers*, 54(3) :340–346, Mar. 2005.

[198] D. Stehlé. *Algorithmique de la réduction de réseaux et application à la recherche de pires cas pour l'arrondi de fonctions mathématiques*. PhD thesis, Université Henri Poincaré-Nancy I, 2005.

[199] E. M. Stein. *Harmonic analysis : real-variable methods, orthogonality, and oscillatory integrals*, volume 43 of *Princeton Mathematical Series*. Princeton University Press, Princeton, NJ, 1993. With the assistance of Timothy S. Murphy, Monographs in Harmonic Analysis, III.

[200] S. Story and P. T. P. Tang. New algorithms for improved transcendental functions on IA-64. In Koren and Kornerup, editors, *Proceedings of the 14th IEEE Symposium on Computer Arithmetic*, pages 4–11, Los Alamitos, CA, Apr. 1999. IEEE Computer Society Press.

[201] G. Tenenbaum. *Introduction to analytic and probabilistic number theory*, volume 163. American Mathematical Society, 2015.

[202] The FPLLL development team. fplll, a lattice reduction library. Available at https://github.com/fplll/fplll, 2016.

[203] S. Torres. *Tools for the Design of Reliable and Efficient Functions Evaluation Libraries*. PhD thesis, École normale supérieure de Lyon – Université de Lyon, Lyon, France, 2016.

[204] L. N. Trefethen. *Approximation Theory and Approximation Practice*. SIAM, 2013. See http://www.chebfun.org/ATAP/.

[205] W. Tucker. The Lorenz attractor exists. *C. R. Acad. Sci. Paris Sér. I Math.*, 328(12) :1197–1202, 1999.

[206] W. Tucker. A rigorous ODE solver and Smale's 14th problem. *Found. Comput. Math.*, 2(1) :53–117, 2002.

[207] W. Tucker. *Validated numerics, A short introduction to rigorous computations*. Princeton University Press, Princeton, NJ, 2011. A short introduction to rigorous computations.

[208] J. D. Vaaler. Some extremal functions in Fourier analysis. *Bulletin of the American Mathematical Society*, 12(2) :183–216, 1985.

[209] B. Vallée. La réduction des réseaux. Autour de l'algorithme de Lenstra, Lenstra, Lovász. *Informatique théorique et applications*, 23(3) :345–376, 1989.

[210] B. Vallée. *Une approche géométrique des algorithmes de réduction des réseaux en petite dimension*. PhD thesis, Université de Caen, 1986.

[211] P. van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. Technical Report 8104, University of Amsterdam, Department of Mathematics, Netherlands, 1981.

[212] L. Veidinger. On the numerical determination of the best approximations in the Chebyshev sense. *Numerische Mathematik*, 2 :99–105, 1960.

[213] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, third edition, 2013.

[214] K. Weierstrass. Über die analytische Darstellbarkeit sogenannter willkürlicher Functionen einer reellen Veränderlichen. *Sitzungsberichte der Akadamie zu Berlin*, pages 633–639, 789–805, 1885.

[215] B. Widrow and I. Kollár. *Quantization Noise – Roundoff Error in Digital Computation, Signal Processing, Control, and Communications*. Cambridge University Press, Cambridge, UK, 2008.

[216] Q. Wu. *Mesure d'indépendance linéaire de logarithmes et diamètre transfini entier*. PhD thesis, Université de Metz, 2000.

[217] Q. Wu. A new exceptional polynomial for the integer transfinite diameter of $[0, 1]$. *J. Théor. Nombres Bordeaux*, 15(3) :847–861, 2003.

[218] Q. Wu. On the linear independence measure of logarithms of rational numbers. *Math. Comp.*, 72(242) :901–911, 2003.

[219] N. Yamamoto. A numerical verification method for solutions of boundary value problems with local uniqueness by Banach's fixed-point theorem. *SIAM Journal on Numerical Analysis*, 35(5) :2004–2013, 1998.

[220] J. Yli-Kaakinen and T. Saramäki. A Systematic Algorithm for the Design of Lattice Wave Digital Filters With Short-Coefficient Wordlength. *IEEE Transactions on Circuits and Systems*, 54(8) :1838–1851, Aug. 2007.

[221] A. Ziv. Fast evaluation of elementary mathematical functions with correctly rounded last bit. *ACM Transactions on Mathematical Software*, 17(3) :410–423, Sept. 1991.

[222] R. Zumkeller. *Global Optimization in Type Theory*. PhD thesis, École Polytechnique, 2008.

[223] A. Zygmund. *Trigonometric series. Vol. I, II*. Cambridge Mathematical Library. Cambridge University Press, Cambridge, third edition, 2002.

Résumé

Dans ce mémoire, j'évoque les travaux de recherche que j'ai menés depuis mon doctorat. L'essentiel de mon activité est consacrée à l'arithmétique des ordinateurs et, plus précisément, à l'évaluation de fonctions en machine. Un objectif général de mon travail est l'amélioration de la qualité (précision, fiabilité, garantie), de la vitesse et du coût (en mémoire, en énergie, en silicium) des calculs en machine. On souhaite pouvoir ainsi produire des logiciels fiables, rapides et portables et du matériel (processeurs, systèmes embarqués par exemple) rapide, fiable et moins coûteux.

Une caractéristique de mon approche de ces questions d'arithmétique des ordinateurs est la mobilisation d'outils de théorie des nombres, plus précisément d'approximation diophantienne, et de calcul formel, plus précisément de calcul symbolique-numérique, pour modéliser et attaquer ces problèmes.

Dans la première et principale partie de ce texte, je présente les travaux que j'ai réalisés, en collaboration, sur l'évaluation des fonctions élémentaires : j'étudie l'approximation sous contraintes sur les coefficients, l'arrondi correct de l'évaluation de la fonction ou encore la certification de l'erreur d'approximation. Les sujets des thèses que j'ai co-encadrées relèvent tous de cette première partie.

Dans la seconde partie, j'évoque brièvement la mise au point d'algorithmes de bas niveau pour accélérer certaines primitives pour le calcul flottant ou la cryptologie à clé publique (couplages sur des courbes elliptiques) et mes travaux en approximation diophantienne.