Calcul numérique certifié dans les espaces fonctionnels : Un trilogue entre approximations polynomiales rigoureuses, calcul symbolique et preuve formelle

Florent Bréhard


Numéro National de Thèse : 2019LYSEN032

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée par
l'Ecole Normale Supérieure de Lyon

Ecole Doctorale N° 512
École Doctorale en Informatique et Mathématiques de Lyon

Spécialité de doctorat : Informatique
Discipline : **Informatique**

Soutenue publiquement le 12/07/2019, par :
Florent BRÉHARD

Certified numerics in function spaces: Polynomial approximations meet computer algebra and formal proof

---

Calcul numérique certifié dans les espaces fonctionnels :
Un trilogue entre approximations polynomiales rigoureuses,
calcul symbolique et preuve formelle

Devant le jury composé de :

Boldo, Sylvie	Directrice de recherche, INRIA	Rapporteure
Wilczak, Daniel	Professeur, Université Jagellonne de Cracovie	Rapporteur
Yap, Chee	Professeur, New-York University	Rapporteur
Beckermann, Bernhard	Professeur, Université de Lille	Examinateur
D'Amico, Simone	Assistant Professor, Stanford University	Examinateur
Brisebarre, Nicolas	Chargé de recherche, CNRS	Directeur
Joldes, Mioara	Chargée de recherche, CNRS	Codirectrice
Pous, Damien	Directeur de recherche, CNRS	Coencadrant

Contents

List of Figures

List of Tables

List of Algorithms

Remerciements

Figure incontournable du manuscrit de thèse, la page des remerciements tient davantage, en termes pâtissiers, de l'empilement multicouche de la Forêt Noire de nos voisins germaniques que du raffinement de nos douceurs hexagonales. Si l'esthète peut légitimement s'en offusquer, l'étudiant en thèse, fraîchement doctorisé, a quant-à-lui à cœur de remercier toutes les personnes qui ont, par le passé, ponctuellement ou tout au long de la thèse, contribué à cet aboutissement. Et elles sont nombreuses.

Je tiens tout d'abord à remercier mes trois directeurs de thèse, d'une part pour l'exceptionnelle qualité de leur encadrement, mais également pour tout ce qu'ils m'ont apporté sur le plan humain, au-delà de tout ce que j'avais pu imaginer avant de me lancer dans cette aventure. La coutume voulant que j'énumère au passage quelques anecdotes, ces trois-là n'y échapperont pas !

Un grand merci à toi Nicolas, pour ton engagement et ton soutien infaillibles, pour toute l'énergie que tu as consacrée à cette tâche, là où tu aurais pu l'économiser pour d'autres combats. Que de chemin parcouru depuis le jour où, te mêlant à je ne sais quelle conversation entre collègues à mon sujet, tu me proposas de découvrir tes thématiques de recherche. Me voici donc docteur, quatre années plus tard ! Le temps passe vite et a déjà eu raison de ce vouvoiement anachronique, ta marque de fabrique qui ne manque pas d'amuser (ou d'agacer !) les autres chercheurs. Au delà de l'incroyable travail accompli ensemble, restent les souvenirs indélébiles de ces *séances de travail* chez toi ou par *Skype*, nos discussions politiques où le président Macron en prit pour son grade, les prises de bec avec l'administration, les éclairs au chocolat achetés en douce à la boulangerie du coin, les déjeuners au *Comptoir du Vin* (incontournable à la Croix-Rousse !), ainsi que tous les chipotages de virgules dans les brouillons d'articles !

Un grand merci à toi Mioara pour l'énergie débordante que tu as déployée pendant quatre ans, ton *inspiration*, ton courage sans lesquels cette thèse ne serait pas le dixième de ce qu'elle est aujourd'hui. Certes, tu n'étais pas particulièrement enchantée de devoir me rencontrer quand Nicolas te le proposa – tu projetas même de refourguer le « normalien, encore un *weirdo* » à d'autres instituts toulousains. C'est pourtant bien toi qui quelques mois plus tard me convainquis de poursuivre en thèse, et qui t'investis à 200% dans ce projet. À côté des journées et des nuits blanches de boulot que cela nous coûta, je ne peux pas ne pas évoquer les dîners à *El Deseo*, les discussions interminables chez toi et Bogdan sur fond de *țuica*/rap roumain/chanson française/tubes américains, les délires sur toutes sortes de sujets moustachus et très peu politiquement corrects, les engueulades pour des broutilles, les débats sur comment les Français devraient prononcer les mots anglais. Sans oublier les deux merveilleux voyages en Roumanie, où je fus si bien accueilli par ta famille !

Un grand merci à toi Damien pour m'avoir fait confiance il y a déjà plus de quatre ans

lors de mon stage de M2 et de m'avoir suivi jusqu'à aujourd'hui. Bien que la thèse se soit davantage orientée vers le calcul rigoureux, tu as réussi tout au long à me faire persévérer dans mon objectif d'aller jusqu'à la preuve formelle, tout en t'ouvrant à ces autres thématiques. Et c'est sans compter sur ta patience que j'ai sans aucun doute mise à rude épreuve lorsque tu reprenais mes scripts Coq fouillis, bien que tu n'aies jamais rien laissé paraître ! C'est une réelle satisfaction pour moi de constater comment nos discussions au tableau ont fini par se concrétiser en un développement Coq fonctionnel. À cela s'ajoute ton sens de la pédagogie et tes précieux conseils.

L'administration de l'ENS fut quelque peu perturbée par cet encadrement à trois ; il eût été indélicat de notre part de surenchérir. Pourtant, Denis, tu as été incontestablement un quatrième encadrant pour moi. Que ce soit pour débloquer les imbroglios administratifs les plus absurdes (comparaison n'est pas raison !), pour m'ouvrir aux thématiques de l'automatique et du spatial, me faire prendre conscience de certaines réalités de notre milieu, ou tout simplement pour discuter de tout et de rien le midi à la cantine (De Gaulle, Bourdieu, la troisième catégorie de personnes après les gentils et les méchants, bien qu'il y en ait d'autres...), tu m'as toujours épaulé comme ton propre étudiant, tout en me considérant comme ton collègue. Ton intégrité restera pour moi un modèle, et tant pis pour les grincheux et autres *snakes*.

Si la rédaction du manuscrit est bien souvent une étape éprouvante pour l'étudiant, sa relecture par les directeurs, puis par les membres du jury l'est tout autant. Je tiens donc à remercier mes rapporteurs Sylvie, Daniel et Chee pour leur intense et minutieux travail de relecture de ces quelques trois cents pages, ainsi que mes examinateurs Bernd et Simone pour s'y être plongés en si peu de jours. Merci à vous tous d'avoir assisté à la soutenance, physiquement ou à distance.

Ma gratitude revient également aux différentes personnes avec qui j'ai eu beaucoup de plaisir à collaborer : Warwick, pour m'avoir fait découvrir le monde des systèmes dynamiques au sein de ton équipe à Uppsala, et pour toutes les discussions autour d'un verre ou au restaurant ; Assia, pour nous avoir apporté ton expertise à Damien et moi, avec une patience et une bienveillance adorables ; Jean-Bernard pour tous les bons moments (de mesures ou de blagues potaches !) ; Paulo et Clément, qui avez eu la curiosité de vous intéresser à mes obsessions de bornes rigoureuses pour l'appliquer à vos thématiques ; Aude, pour m'avoir toujours chaleureusement intégré à tes travaux avec Denis et Mioara.

S'ajoutent à elles les très nombreuses personnes que j'ai eu la chance de côtoyer à Lyon et à Toulouse, au sein des trois équipes qui m'ont accueilli : AriC, Plume et MAC. Chacun, par sa personnalité, ses connaissances et son humour, a contribué à ce que chaque journée de travail ait son lot de surprise et de réjouissance. S'il m'est impossible ici d'être exhaustif, je m'autorise tout de même à citer quelques noms : Bruno, pour tes connaissances D-finies et ta pédagogie infinie ; Jean-Michel, pour ta bienveillance, tes conseils et tes blagues ; Gilles, pour ton écoute et ton humour ; Serge, pour ton expérience, ta sagesse et ton humanité ; Joris, pour ton implication dans des thématiques qui me sont chères, en parallèle du support technique que tu fournis à l'ensemble du labo ; Nicolas, pour tes plaisanteries et tes questions existentielles sur Coq ; Philippe, pour notre travail d'enseignement ensemble et nos discussions à côté ; Daniel, pour toute ton aide et tes conseils sur l'enseignement ; Guillaume puis Patrick, pour votre soutien en tant que directeur du LIP ; Fred, pour tes conseils mais aussi toutes les rigolades dans ton bureau au LAAS, entre hyènes et chacals !

Je tiens tout particulièrement à remercier les doctorants de ces différentes équipes. Notamment, dans AriC : Alice, pour tes gâteaux et tes vannes ; Fabrice, pour tes délires et éclats de rire ; Valentina, pour tous ces souvenirs à Lyon et à Cavnic, Ida, pour nos joggings-papotage

dans le froid de l'hiver lyonnais ; Miruna et Radu, pour avoir pris le relais de Valentina dans mes cours de roumain, Anastasia, pour avoir toujours été si gentille et joyeuse ; Alex, bon camarade actuellement exilé chez les Nippons. Chez les logiciens de Plume : Aurore, pour ta générosité infaillible, ta finesse et tes rires discrets, Pierre, Adrien et Marc, compagnons de galère pour les TD, et camarades indéfectibles pour discuter de tout et n'importe quoi : les monades, Coq et le trotskysme ; Laureline, pour ton sens du collectif mais aussi du rire. Petit détour par MC2 avec toi Alex, bien que tu n'aies pas réussi à me faire venir dans le club jeux. Enfin chez les automaticiens de MAC : Paulo et Clément, en souvenir de l'escapade lyonnaise dans les bouchons, Flavien, pour ta sollicitude pendant nos quelques séances de sport, et nos franches rigolades serpentines, Matthieu, pour être toujours resté à l'écoute des autres malgré un agenda particulièrement chargé cette année.

Si j'ai pu travailler dans d'aussi bonnes conditions tant à Lyon qu'à Toulouse, je le dois avant tout au soutien qui m'a été apporté face aux lenteurs et pénibilités de l'administration. Un immense merci à vous deux, Marie et Chiraz, pour m'avoir épaulé tant de fois dans mes diverses démarches (convention de codirection, contrat avec le CNES, etc.), mais aussi pour avoir été si gentilles et m'avoir fait tant rire en trois ans : votre présence au LIP est précieuse. Merci à toi Nelly pour m'avoir tant aidé lors de mes (nombreuses) missions. Merci à toi Jean-Christophe pour avoir si bien géré les aspects techniques de ma soutenance. Merci à toi Cathy pour toute l'aide que tu m'as apportée lors de mes séjours toulousains, pour ta gentillesse et ta bonne humeur qui ne t'ont jamais fait défaut.

En plus de ces deux laboratoires, je remercie également le CNES pour m'avoir accueilli et permis de travailler sur des thématiques passionnantes, en particulier Sophie pour avoir organisé ce séjour et préparé le terrain ; Pierre pour m'avoir toujours aidé et suivi avec écoute durant ces semaines ; Lætitia et Doriane pour m'avoir apporté un soutien administratif des plus bienveillants.

Plus loin dans le temps ou dans l'espace, je tiens aussi à souligner le rôle qu'ont joué d'anciens collègues et encadrants lors de mes premières expériences de recherche : Laurence, Yves, Laurent et les autres membres de Marelle à Sophia Antipolis, pour m'avoir offert ma première expérience de chercheur, et mes premiers contacts avec Coq et la théorie homotopique des types ; Pierre, Tom, Raphael, Rodolphe, Marion, pour cet inoubliable stage à Chambéry où je fus si bien accueilli ; Olivier, pour avoir dû corriger à plusieurs reprises mes preuves d'élimination des coupures en M2 ; Jordi, Denis et les autres membres de l'équipe CAPA pour toute leur aide lorsque Warwick m'a invité à Uppsala ; Laura, Marco, Mariano, Azul et César pour m'avoir si bien accueilli au NIA en Virginie pour mon premier voyage transatlantique.

Ces trois années ont aussi pour moi été l'occasion de rencontrer des personnes formidables en dehors du boulot. Je ne sais comment assez te remercier, Bogdan, en premier lieu pour avoir supporté avec tant de flegme nos chamailleries avec Mioara, ainsi que les nuits blanches pour cause de *deadline* pendant lesquelles tu nous a toujours soutenus, cuisinant au besoin une basse-côte juste saisie pour nous redonner du courage. Merci à l'expert *ès* vélos, à l'auteur de jeux de mots improbables, au sauveur du chat noir (ou plutôt du papa !), toutes ces qualités qui font de toi un type génial !

Côté lyonnais, merci également à Nathalie et la petite Juliette de m'avoir toujours accueilli de si bon cœur, que ce soit pour les séances de travail avec Nicolas à la Croix-Rousse, ou pour des dîners toujours excellents et chaleureux ! Merci aussi à Olivier et Paul pour toutes ces soirées Mario Kart endiablées, dans une bonne humeur qui illumina mes hivers à Lyon.

Si cette période de ma vie restera aussi une des plus heureuses, je le dois sans aucun doute à mes fantastiques expériences de colocation. J'adresse mes remerciements les plus tendres à

l'ensemble de l'auberge de la Cocagne à Toulouse, qui, si elle ne comporte que neuf membres simultanément, a vu défiler un très grand nombre de personnes dont je m'efforce ici de reproduire la liste exhaustive : Louise, Naïm, Lorette, Val, Morgane, Baptiste, Marie, Anaïs, Vincent et Hermance, Marie, Hanaé et Thomas, Camille et Jérem, Thomas, Amaïa, Thibault et Camillou, Valou, Clément, Laury, Clara, Rémy, Arnaud, Béré et Manu, Laura, Baptiste, Seb, Margaux, Marianne, Pierre. À chacun de vous, un immense merci pour m'avoir apporté tant de bonheur. Merci également aux colocs avec qui j'ai eu la chance de vivre à Lyon : Élodie et Julie (et Louis !) pendant deux ans, puis Frankie et Sarah la dernière année. Merci aussi à tous mes amis et anciens camarades de scolarité. Si la thèse a été peu propice à des retrouvailles régulières, je n'oublie pas pour autant ce que vous m'avez apporté.

Nous conclurons cette recette par les fondements du gâteau, ceux sans lesquels ce chemin n'aurait pas été possible. Un immense merci à toute ma famille qui m'a toujours soutenu dans ma curiosité et encouragé dans mes efforts, en particulier mes parents, ma sœur Romane, mes oncle et tante Thierry et Murielle, mes cousins Pierre et Camille. Plus récemment, merci pour votre soutien pendant ces trois années intenses et pour avoir été présents jusqu'à la fin : la soutenance ! Je remercie également l'ensemble de mes professeurs au cours de ma scolarité, et en profite pour exprimer ma gratitude la plus profonde envers mon institutrice Martine Gadon (alias *Maîtresse*). Tu as eu confiance en mes capacités et ma soif de connaissances dès mon plus jeune âge. Ton jeune élève est aujourd'hui docteur !

Introduction

– Si tu veux un ami, apprivoise-moi !
– Que faut-il faire ? dit le petit prince.
– Il faut être très patient, répondit le renard. Tu t'assoiras d'abord un peu loin de moi, comme ça, dans l'herbe. Je te regarderai du coin de l'œil et tu ne diras rien. Le langage est source de malentendus.
— Antoine de Saint-Exupéry, Le Petit Prince

Numerical analysis underlies a wide range of computational methods involving problems with continuous data (time, space, etc.), with applications to, for instance, engineering simulations, image processing or decision making procedures. A main achievement is the development of highly optimized implementations for various numerical algorithms in linear algebra, functional analysis or continuous optimization. These implementations greatly benefit from the increasing efficiency of a standardized floating-point arithmetic (via the IEEE 754 standard) on modern computing processors and architectures.

Nevertheless, numerical errors constitute an inherent and much studied problem which is ubiquitous in scientific computing. Two classes of such errors are usually identified. First, *rounding errors* are intrinsic to floating-point arithmetic, since the latter is only an approximation (more precisely, a discretization) of the real line. Second, the *method errors* arise when infinite-dimensional problems are approximated by finite-dimensional ones, thanks to discretization or projection methods. Typical examples include Runge-Kutta or spectral methods to solve ordinary differential equations (ODEs). While usually small, these errors accumulate during the program run and may lead to disastrous errors at the end. In most applications, this simply leads the program to fail in providing an expected accurate answer. The main issue is that often, one cannot directly estimate what is the returned accuracy, that is, how many (or if any) digits of the returned result are correct. Thus, domains with stronger reliability requirements, like safety-critical engineering or computer-assisted proofs in mathematics, cannot rely only on traditional numerical analysis algorithms.

Main objectives. In regard of these shortcomings, this thesis addresses the following challenges. Not all of them will be equally treated in this manuscript – obviously, the first receives the most attention – but each one brings its own light. We list them below and then provide further details on how they are handled, insisting on several key concepts, relevant examples and the outline of this document.

A. Computing faster, more accurate, more reliable, safer. This is the primary goal of this thesis. After a subjective survey of the existing techniques in numerical, symbolic and rigorous computation (**Chapters 1** and **2**), we enhance *rigorous polynomial approximations* (RPAs) with new algorithms, where *a posteriori fixed-point validation* methods play an important role (**Chapter 3**). This brings new efficient and accurate tools for function space problems, notably linear differential equations (**Chapters 4** and **5**). Moreover, we do not only target *validated numerics*, but also *certified numerics* with the use of formal proof (**Chapter 3**).

B. Prove mathematical theorems in new computer-assisted ways. Drawing on an ongoing work concerning limit cycles of a polynomial vector field (**Chapter 6**), we advocate the use of RPAs, a posteriori validation and symbolic-numeric methods in the design of efficient tools for computer-assisted proofs in mathematics. Taking advantage from our a posteriori or *certificate-based* approach, formal proof methods are easier to introduce in the computer-assisted proof process, thus offering the highest level of confidence.

C. Reliability, safety and efficiency for real-life applications. More reliable computations are crucial in safety-critical engineering applications, yet efficiency cannot be sacrificed (too much). By computing and using validated spacecraft trajectories in space proximity operations (**Chapter 7**), we illustrate how our methods can be used in a safety-critical context, like the rigorous space mission design.

D. Taking into account "lower-level" aspects is essential when designing efficient tools and implementations for "real-life" applications. Although this is not the most widely discussed point of this manuscript, this aspect was also investigated. A contribution in that sense is a new exchange algorithm to design evaluation and approximation error optimized polynomials (**Chapter 8**). This is crucial for the numerical implementation of functions, since the underlying floating-point precision is finite.

E. New outlets for rigorous and symbolic-numeric methods. Exporting rigorous and symbolic-numeric techniques to other fields of mathematics or engineering is an interesting challenge, which also gives the opportunity to enhance our tools with new mathematical concepts. For example, the support and density reconstruction for measures from moments presented in **Chapter 9** makes use of *Ore polynomials* – a symbolic tool – in the context of an inverse problem.

F. Designing open-source implementations. Last but not least, software implementations are an essential contribution to this thesis, since they concretely attest our claim to design efficient algorithmic methods. Moreover, they were a valuable aid in the above-mentioned applications.

A Computing faster, more accurate, more reliable, safer

The first five chapters of this manuscript are motivated by this goal. As a preliminary remark, we identify the following three *safety levels* of computation:

1. A first *purely numerical* level encompasses the traditional numerical algorithms, focusing on efficiency and accuracy. Although usually provided with asymptotic convergence estimates, these algorithms *cannot* rigorously guarantee the result's accuracy in general.

2. A second level is dedicated to *validated* algorithms, in a broader sense, meaning that the computed result comes with guarantees of some mathematical properties. This includes:

   - *symbolic computation* (**Goal A.1**) for merely algebraic problems, with *exact representations* of objects and sometimes also *certificates* making it possible to check the result a posteriori;
   - *rigorous numerics* (**Goal A.2**) based on numeric *set-valued* representations of objects, provide enclosures for the exact but maybe not machine-representable result.

   Sometimes, both may even be combined into so-called *symbolic-numeric* methods.

3. A third *certified* level offers the highest safety, namely at implementation level. *Formal proof* (**Goal A.3**) allows us to design *certified* implementations, whose correctness is guaranteed by highly trusted interactive theorem provers.

Determining the adequate safety level is really an application-dependent choice: are we ready to spend more time on a specific computation for a higher confidence in the result? However, it is also limited to existing tools and software. In particular, certified implementations of numerical methods are still rare and less efficient than their nonrigorous analogs.

Before addressing the contributions of this thesis, an overview of the necessary computational tools is given by the first two chapters. **Chapter 1** discusses computer representations of numbers: exact representations for symbolic computation; floating-point numbers, highly efficient but subject to rounding errors; and intervals to rigorously enclose a real number. Then, **Chapter 2** tackles representations of functions, focusing on two particular topics: the symbolic framework of D-finite functions, that is, solutions of linear ordinary differential equations (LODEs) with polynomial coefficients, and a condensed summary of polynomial approximation theory, which is useful for the following chapters.

Rigorous polynomial approximations (RPAs), widely investigated throughout this thesis, are presented in **Chapter 3**, together with *a posteriori fixed-point validation* methods, which we use to define fixed-point based division and square root of RPAs. The resulting arithmetic on RPAs is the foundation of more elaborate rigorous methods for function space problems. A central contribution of this thesis is algorithms to compute RPAs in Chebyshev basis for LODEs solutions, conceived with Nicolas Brisebarre and Mioara Joldes. They are presented in **Chapters 4** (for scalar LODEs) and **5** (for coupled systems of LODEs).

The following example, based on the Airy function Ai, will illustrate how the subgoals detailed below also provide complementary viewpoints.

Example 1 : Airy function Ai

The Airy function Ai (see **Figure I.4a**) is a well-known special function in mathematics **[2]**, which can be defined by the following improper Riemann integral:

$$\mathrm{Ai}(x) = \frac{1}{\pi} \int_0^{+\infty} \cos\left(\frac{t^3}{3} + xt\right) \mathrm{d}t, \tag{Ai-i}$$

or by the following second order differential equation with initial conditions at 0, where Γ denotes the Gamma function:

$$\mathrm{Ai}''(x) - x\,\mathrm{Ai}(x) = 0, \qquad \mathrm{Ai}(0) = \frac{1}{3^{\frac{2}{3}}\Gamma\left(\frac{2}{3}\right)}, \qquad \mathrm{Ai}'(0) = -\frac{1}{3^{\frac{1}{3}}\Gamma\left(\frac{1}{3}\right)}. \tag{Ai-ii}$$

This function does not admit any simpler closed-form formula involving elementary functions, such as exp, sin, cos or their reciprocals. One of the objectives of numerical analysis, computer algebra and rigorous numerics is to develop methods to manipulate such a function: evaluation, verification of identities, validated approximations, etc.

A.1 Symbolic tools

A straightforward solution to avoid numerical errors is using symbolic methods [257], which reflect pen-and-paper mathematics, that is, computing with *exact representations* of numbers, functions, etc. Decades of research in that area gave rise to efficient algorithms with thorough complexity studies, implemented inside user-friendly *computer algebra systems* (CAS) such as Maple or Mathematica. *D-finite functions* are a striking example of such exactly representable objects. They are presented below since they will occur on many occasions in this thesis.

D-finite functions are the solutions of linear ODEs with polynomial coefficients [234]. They are ubiquitous in mathematics – they represent about 60% of the functions listed in [2], e.g., the Airy function Ai of **Example 1**. Thanks to the efforts of an active community in symbolic computation, computer algebra systems like Maple or Mathematica make it possible to represent such functions as a *data structure* (a differential operator annihilating the function plus sufficiently many initial conditions), and to operate on them with arithmetic and differential operations.

Example 1 : Airy function Ai – Symbolic manipulation

We illustrate how the algorithmic treatment of D-finite functions can prove nontrivial statements.

1 (a) *Prove the following identity* [2, Eq. (10.4.15)]*:*

$$\mathrm{Ai}(-x) = \sqrt{\frac{x}{9}}\left(J_{\frac{1}{3}}\left(\frac{2}{3}x^{\frac{3}{2}}\right) + J_{-\frac{1}{3}}\left(\frac{2}{3}x^{\frac{3}{2}}\right)\right), \qquad x > 0, \tag{Ai-iii}$$

where the Bessel functions $J_{\pm\frac{1}{3}}$ form a basis of the solutions of:

$$9z^2 f''(z) + 9z f'(z) + (9z^2 - 1)f(z) = 0. \tag{$J_{\frac{1}{3}}$}$$

Using closures operations, implemented as *algorithms*, the Gfun[1] package for Maple automatically *computes* that the right-hand side of (Ai-iii) satisfies $g''(x) + xg(x) = 0$. Indeed, this expression is obtained from $J_{\pm\frac{1}{3}}$ by an algebraic substitution $z \mapsto \frac{2}{3}x^{\frac{3}{2}}$, followed by a sum, and

[1] `http://perso.ens-lyon.fr/bruno.salvy/software/the-gfun-package/`

finally multiplied by $x \mapsto \sqrt{\frac{x}{9}}$. Now, from (Ai-ii), $x \mapsto \mathrm{Ai}(-x)$ also satisfies the same LODE, so that checking equality for initial conditions at 0 is sufficient to establish the identity (Ai-iii), by the Picard-Lindelöf theorem.

Creative Telescoping refers to techniques **[269, 59, 141]** that were developed to automatically compute differential equations or recurrence relations for integrals or sums of D-finite quantities, thus allowing sometimes for non trivial closed forms. Moreover, these somehow complex techniques also provide a certificate that allows for an a posteriori verification of the result. More details will be provided in **Chapter 2**.

Example 2: Exponential generating function of Chebyshev polynomials

2 (a) *Prove the following identity:*

$$\sum_{n=0}^{+\infty} T_n(x)\frac{t^n}{n!} = e^{tx}\cosh(t\sqrt{x^2-1}), \qquad |x| < 1,$$

where the T_n are the Chebyshev polynomials defined in Chapter 2.

The integrand $f(t,n) := T_n(x)\frac{t^n}{n!}$ satisfies the differential-difference equations:

$$t\partial_t \cdot f - nf = 0,$$
$$(2+3n+n^2)S_n^2 \cdot f + (-2tx - 2ntx)S_n \cdot f + t^2 f = 0.$$

where $\partial_t \cdot f(t,n) := \frac{\partial f}{\partial t}(t,n)$ and $S_n \cdot f(t,n) := f(t,n+1)$.

Using, e.g., the `CreativeTelescoping` function from the MATHEMATICA package HOLONOMICFUNCTIONS[2], one obtains the following differential equation for $g(t) := \sum_{n=0}^{+\infty} f(t,n)$:

$$\partial_t^2 \cdot g - 2x\partial_t \cdot g + g = 0.$$

Finally, standard differential equation solving procedures of MAPLE or MATHEMATICA recover the closed-form expression $e^{tx}\cosh(t\sqrt{x^2-1})$ from this differential equation.

Shortcomings of symbolic methods. However, some problems are by essence intractable with such tools, due to the following limitations:

- Computable exact representations are available only for restricted classes of numbers or functions, like rational or algebraic numbers/functions. But consider for example the Euler constant $\gamma := \lim_{n\to\infty}\left(\sum_{k=1}^{n}\frac{1}{k} - \log n\right)$: how are algorithms supposed to compute with it or decide properties on it? – it is not even known whether γ is rational or not.
- Some objects are carried out via exact but implicit representations. Hence, when concrete properties have to be checked, a rigorous numerics step may be necessary to convert the implicit representation into actual numerical values. This is for example the case for D-finite functions, for which concrete rigorous approximations are given in **Chapters 4** and **5** (see below).

[2] `https://www3.risc.jku.at/research/combinat/software/ergosum/RISC/HolonomicFunctions.html`

- Even when exact representations are available, their intrinsic size may be a major obstacle to the practical implementation on a computer. This is often the case for algorithms iterating algebraic operations on exactly represented objects. We will illustrate this phenomenon in **Chapter 1**, **Example 1.3** with the inversion of matrices with rational coefficients.

A.2 Rigorous numerics and validated methods

In this regard, rigorous numerics **[247]** overcome the limitations of symbolic computation by replacing *exact* representations with *validated* ones, for solutions of various problems, notably in functional analysis, e.g., differential equations or optimal control. The key idea is to design computable set-valued representations, such as real intervals or balls in function spaces, with the guarantee that the exact solution of the problem under consideration is contained in it. This relaxation allows one to use floating-point arithmetic and numerical algorithms, and yet to remain correct thanks to a careful use of, e.g., directed roundings or a posteriori bounds.

Interval arithmetic is an essential building block of rigorous numerics (cf. **[173, 247, 221]**), and will be presented in **Chapter 1**. The key idea consists in using real intervals with representable endpoints (e.g., floating-point numbers) as rigorous enclosures of real numbers, and providing operations preserving correctness. For example, from $\pi \in [3.1415, 3.1416]$ and $e \in [2.7182, 2.7183]$, one obtains:

$$\pi + e \in [3.1415, 3.1416] \oplus [2.7182, 2.7183] = [5.8597, 5.8599].$$

Therefore, in principle replacing all floating-point operations by interval ones resolves the issue of rounding errors. However, as detailed in **Chapter 1**, several limitations should prevent us from seeing interval arithmetic as the silver bullet of rigorous numerics:

- Overestimations are a well-known shortcoming of interval arithmetics **[221]**, due to several phenomena such as the *dependency phenomenon* or the *wrapping effect*: the resulting interval is correct, but too large to provide relevant information.

- Although the resulting interval is correct with respect to rounding errors, method errors are not taken into account, since they result from the discretization of the problem itself, not from the approximation of real numbers by floating-point ones.

Rigorous polynomial approximations (RPAs) were conceived as a *higher order* counterpart to interval arithmetic to tackle the dependency phenomenon. They consist in a polynomial together with a bound for the total error between the polynomial and the function it represents, with respect to a given norm (see **Figure I.1**). One of the first conceptualizations of these ideas dates back from the 80s with *ultra-arithmetic* **[79, 80]**. Somehow later, the implementation of *Taylor models* **[164, 165]** (which can be seen as rigorous truncated Taylor series expansions) in the COSY Infinity software revived the interest for RPAs.

However, **Chapter 2** highlights the limits of Taylor expansions in approximation theory, while *generalized Fourier series expansions* – in particular Chebyshev expansions – are often far more efficient. In view of this observation, the *Chebyshev models*, introduced in **[44, 129]**, were rapidly adopted by some computer-assisted proofs in dynamical systems (see, e.g., **[153]**).

Moreover, the fully algorithmic approach proposed for Chebyshev models (e.g., [19] for D-finite functions) is an important source of inspiration for this thesis.

Chapter 3 provides an arithmetic for Chebyshev models, with some differences with [129], notably for division and square root that are now treated using an "*interpolation – a posteriori validation*" approach. A C library CHEBVALID[3] for Chebyshev models as well as a COQ formalization[4] of RPAs are also presented in this chapter (see **Goal F** below). **Chapter 4** also relies on *a posteriori fixed-point validation* (see below) to automatically compute RPAs for solutions of LODEs with coefficients represented by RPAs. Contrary to [19], which is limited to D-finite functions, the proposed approach is closer in spirit to the Newton-like validation methods, presented below.

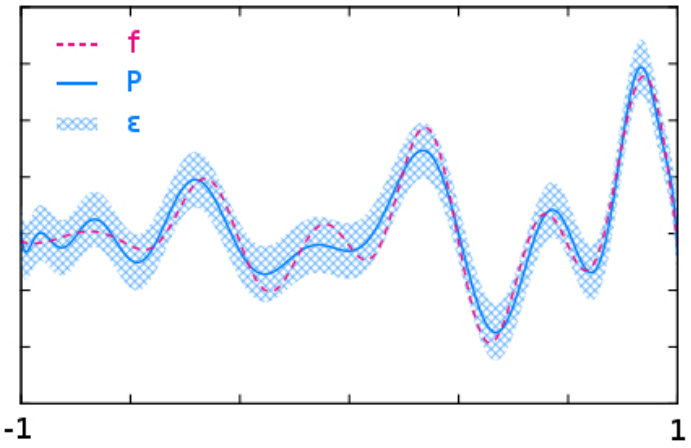


Figure I.1: An RPA for the spacecraft rendezvous problem: polynomial approximation + rigorous error bound (in blue) form a tube containing the exact trajectory (dashed magenta).

A posteriori fixed-point validation methods are a very powerful tool for rigorous numerics when *direct* (or *self-validating*) techniques are not available or sufficiently accurate. The problem is solved in two independent steps:

1. The *approximation step* computes an approximation x° of the exact solution x^* of the problem under consideration. Any numerical algorithm can be used – no hypothesis is needed.

2. The *validation step* rephrases the initial problem to make x^* the unique fixed point of a well-chosen contracting operator, and a rigorous upper bound on the error of x° to x^* is afterwards reconstructed using the *Banach fixed-point theorem*.

This two-step approach, presented in more details in **Chapter 3**, is widely used in computer-assisted proofs for function space problems and dynamical systems, notably via *a posteriori Newton-like validation methods* (see, e.g., [137, 136, 178, 199, 189, 265, 9, 250, 153, 111], or the survey [179] and references therein). Contrary to the mainstream *case by case* strategy

[3] available at `https://gforge.inria.fr/projects/tchebyapprox/`
[4] available at `http://perso.ens-lyon.fr/florent.brehard/chebapprox/`

of these works, the validation procedure for LODE solutions presented in **Chapter 4** provides a fully algorithmic approach with detailed complexity estimates, that was moreover implemented in the above-mentioned ChebValid C library.

Still on the topic of a posteriori validation, **Chapter 5** gives a new generalization of the Banach fixed-point theorem, allowing us to provide a general framework for tight componentwise validation of vector-valued problems. This is then applied to the validation of *coupled systems of LODEs*, using the techniques of **Chapter 4**.

Example 1 : Airy function Ai **– Rigorous approximation**

1 (b) *How to design efficient RPAs for* Ai*?*

The Airy function Ai discussed above in the context of symbolic manipulation, can now be rigorously approximated and evaluated using Chebyshev models. **Section 4.5.1** gives a detailed account of how the validation method for LODEs presented above computes and validates approximations of Ai. Note that, instead of the Chebyshev approximations provided by **Chapter 4**, one can also apply this validation method to any polynomial, in particular the evaluation error optimized polynomials computed in **Chapter 8** (see **Goal D**).

A.3 Formal proof and certified implementations

Formal proof may be used to offer the highest level of confidence for rigorous numerics, that is, certified implementations. *Proof assistants* (also named *interactive theorem provers*) are software in which the programmer edits *proofs* for algorithms or mathematical statements. These proof scripts are then checked line by line by the kernel, i.e. a usually rather small and highly trusted codebase on which all the correctness of this verification step relies. Examples of famous proof assistants are: Mizar, ACL2, HOL4, HOL-Light, Isabelle, Coq, PVS, Lean, etc. Although many of them provide some automation tools (e.g., tactics in Coq or the Sledgehammer toolbox for Isabelle), formal proof tends to be a time-consuming task requiring a large workforce. In return, some formalizations of important theorems/conjectures received a wide recognition and reinforced the place of computer-assisted proofs among mathematicians (see **Goal B** below).

In the following, I will focus on the Coq proof assistant[5] **[22]** developed at INRIA, which is the one that we used for our formalization of RPAs (see **Goal F**). The *type theory* underlying Coq's logic is briefly presented below.

Type theory and proofs. Typed programming languages classify data into different predefined or user-defined categories, called *types*; the notation `a : A` stands for "the term `a` has type `A`". For example, in Coq, `nat` stands for the nonnegative integers, `bool` for the Booleans, `A → B` for the functions taking an argument of type `A` and returning a value of type `B`, etc. The *Curry-Howard correspondence* **[92,** Chap. 3**]** makes it possible to see types as mathematical propositions, and terms of that type as proofs of the corresponding proposition. For instance, a proof of the mathematical implication $A \Rightarrow B$ is nothing more than a function of type `A → B`, constructing a proof of B out of a proof of A.

[5] `https://coq.inria.fr/`

The Coq proof language enhances this correspondence using *dependent types* [169][22, Chap. 4]. A dependent type is simply a type $B(a)$ parametrized by inhabitants a of the type A, that is a term `B : A → Type`. The *dependent product* `∀ (a : A), B a` generalizes the usual arrow type; it contains functions f that associate, to any argument `a : A`, a return value `f a : B a`. Clearly, dependent products are the analog of the universal quantifier in mathematical logics (and similarly, there are *dependent sums* for the existential quantifier). Finally, Coq also allows the user to define *inductive types* [22, Chap. 6], whose elements are constructed out of a finite set of constructors declared by the user. For instance, the standard definition of nonnegative integers in Coq is the following inductive type, corresponding to a unary representation of numbers (see **Chapter 1**):

```
Inductive nat : Set :=
    | O : nat
    | S : nat → nat.
```

Then Coq automatically associates an *induction principle* to it that roughly says: "All the possible patterns with `O` and `S`, that is `S (S (... (S O)))`, are distinct terms of `nat`, and all the terms of `nat` have this shape".

```
nat_rect : ∀ P : nat → Type, P O → (∀ n : nat, P n → P (S n)) → ∀ n : nat, P n.
```

We shall stop here this short introduction to type theory ad refer to [92, 22] for more details. In fact, type theory can be used as logical foundations of mathematics, in replacement of usual *set theory*. For example, the *Univalent Foundations Program*[6] aims at redefining all the mathematics using an extension called *Homotopy Type Theory* (HoTT) [204]. In the following, we focus more specifically on formalization of mathematical analysis in the Coq proof assistant.

Analysis in Coq is a widely debated topic in the formal proof community. A major reason is that in Coq's purely constructive logic, the different usual constructions of the field of real numbers (e.g., Dedekind cuts or Cauchy sequences) are not equivalent. Moreover, a purely constructive definition of real numbers necessarily lacks important properties, such as the *zero test* `∀ (x : R), x = 0 \/ x <> 0`. Therefore, the Coq standard library opted for an axiomatized type `R` for real numbers, but constructive alternatives can still be used, such as the C-CoRN/MathClasses library [63].

Based on the above-mentioned axiomatization of real numbers, the Coquelicot library [29] provides a user-friendly framework for real and functional analysis. Our formalization of the Banach fixed-point theorem, that we use for the fixed-point based operations on RPAs, strongly relies on Coquelicot's topological foundations via filters, as detailed in **Chapter 3**.

On the numerical analysis side, the Flocq library [30] formalizes floating-point arithmetic, which in turn is used by interval arithmetic, formalized in the CoqInterval library [171]. This allows for the automated solving of real inequalities using interval analysis. Closer to our work, CoqApprox [168] formalizes Taylor models, thus enhancing the features of CoqInterval.

B Prove mathematical theorems in new computer-assisted ways

[6] `https://homotopytypetheory.org/`

Computer-assisted proofs consist in delegating part of the proof of a mathematical theorem to the computer. Several reasons may advocate this somewhat daring choice: too intricate computations for pen-and-paper work, combinatorial explosion of cases to be analyzed, very high level of abstraction making proofs doubtful, etc. The personal experience of V. Voevodsky, extracted from his note "The Origins and Motivations of Univalent Foundations"[7], particularly highlights the limits of classical pen-and-paper mathematics:

> *I can see two factors that contributed to this outrageous situation: Simpson claimed to have constructed a counterexample, but he was not able to show where the mistake was in our paper. Because of this, it was not clear whether we made a mistake somewhere in our paper or he made a mistake somewhere in his counterexample. Mathematical research currently relies on a complex system of mutual trust based on reputations. By the time Simpson's paper appeared, both Kapranov and I had strong reputations. Simpson's paper created doubts in our result, which led to it being unused by other researchers, but no one came forward and challenged us on it.*

Computer-assisted proofs in analysis make use of floating-point arithmetic and numerical algorithms, yet they have to guarantee mathematical properties about the computed results – a theorem cannot be true *up to* numerical errors! To this aim, an essential ingredient is the *rigorous*, or *set-valued* numerics introduced in **Goal A.2**. They played a major role in several iconic computer-assisted proofs, such as Landford's proof of the Feigenbaum conjecture **[147]**, Tucker's proof about the existence of the Lorenz attractor **[246]**, or Hales' proof of the Kepler conjecture **[102]**. Dedicated libraries for rigorous numerics were also developed toward computer-assisted proofs, such as the CAPD library[8] for dynamical systems theory.

Of course, delegating some parts of the proof to a computer is absolutely not a minor decision, and computer-assisted proofs are sometimes called into question by some mathematicians, who may doubt the rigor of such approaches, or refuse to believe in a result they cannot check by hand line by line. For the former point [9], the use of *formal proof assistants* should definitely close down the debate, like the four color theorem **[95]** by Gonthier and colleagues in Coq, the Flyspeck project **[101]** by Hales and colleagues in HOL-Light for the Kepler conjecture, or the formalization by Immler **[119]** in Isabelle of Tucker's computer-assisted proof for the existence of the Lorenz attractor.

The joint work with Nicolas Brisebarre, Mioara Joldes and Warwick Tucker, described in **Chapter 6**, is an example of computer-assisted proof in analysis, using first rigorous, and then formally certified computations. Hence, we used both the ChebValid C library and Coq development (detailed in **Goal F**) toward the objective summarized in the following example.

Example 3 : Computing limit cycles in Hilbert's 16th problem

Chapter 6 of this manuscript consists in computing limit cycles in the framework of Hilbert's 16th problem:

> *What is the maximum number $\mathcal{H}(n)$ of limit cycles a polynomial vector field of degree n*

[7] `https://www.ias.edu/ideas/2014/voevodsky-origins`
[8] `http://capd.ii.uj.edu.pl/`
[9] For the latter, however, it is merely a question of personal preference – and, after all, you are not obliged to like a proof to admit it.

in the plane can have, that is:

$$\begin{cases} \dot{x} = P(x, y) \\ \dot{y} = Q(x, y) \end{cases}$$

with $P, Q \in \mathbb{R}[x, y]$ *of degree at most* n*?* ”

More detailed definitions are provided in the dedicated chapter. Roughly speaking, a *limit cycle* is an isolated periodic orbit, attracting sufficiently close trajectories (in positive or negative time). A well-known example is given by the Van der Pol oscillator depicted in **Figure I.2a**.

We do not address Hilbert's 16th problem itself, but instead constructs an example to compute a new *lower bound* for $\mathcal{H}(4)$. In fact, our starting point was the article **[126]**, in which T. Johnson claims to rigorously isolate 26 limit cycles for a quartic system he built on purpose. Based on the Poincaré-Pontryagin theorem, which relates the limit cycles to the zeros of so-called *Abelian integrals*, the proof consists in rigorously evaluating these integrals to observe sign alternations. Unfortunately, the rigorous software designed toward this goal was erroneous, and the correct number of limit cycles was 18, instead of 26 (which is less than the previous lower bound $\mathcal{H}(4) \geqslant 22$ **[56]**).

Our motivation was to apply the rigorous techniques developed in this thesis to fix T. Johnson's example and obtain as much zeros as possible – rigorously, of course!

3 (a) *Find a concrete example of a quartic vector field and isolate limit cycles to obtain a new and rigorous lower bound for* $\mathcal{H}(4)$*.*

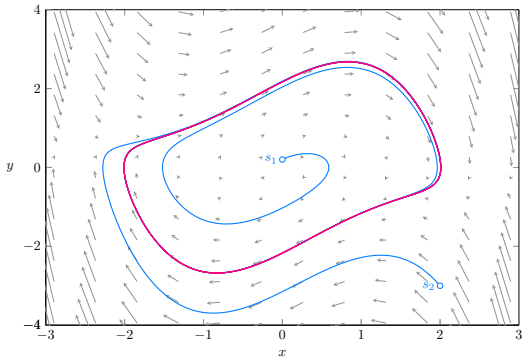
To do this, we reused the example of T. Johnson, depicted in **Figure I.2b**:

$$\begin{cases} \dot{x} = -4y^2(y^2 - Y_0) + \varepsilon g_1(x, y), \\ \dot{y} = \quad 4xy(x^2 - X_0) + \varepsilon g_2(x, y), \end{cases}$$

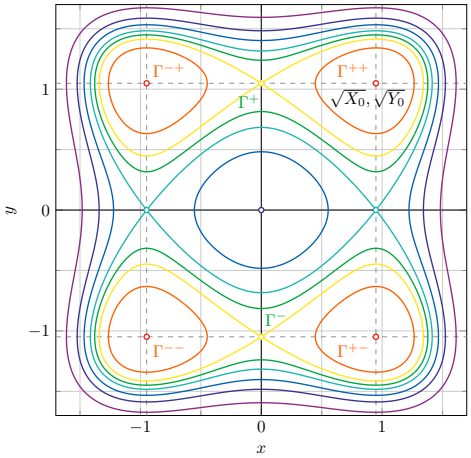
with $\varepsilon > 0$ and tuned the coefficients of the polynomial perturbation $(f, g) \in \mathbb{R}_4[x, y]^2$ to obtain 24 zeros, which therefore improves the previous record $\mathcal{H}(4) \geqslant 22$. Evaluating the Abelian integrals amounts to integrating algebraic functions over a compact segment, which is possible thanks to RPAs, implemented in ChebValid. **Figure I.2c** depicts the rigorous evaluation of the Abelian integral, whose sign alternations rigorously prove the existence of the desired limit cycles.

3 (b) *Can this approach be considered as a* valid *proof of* $\mathcal{H}(4) \geqslant 24$*, in the mathematical sense?*

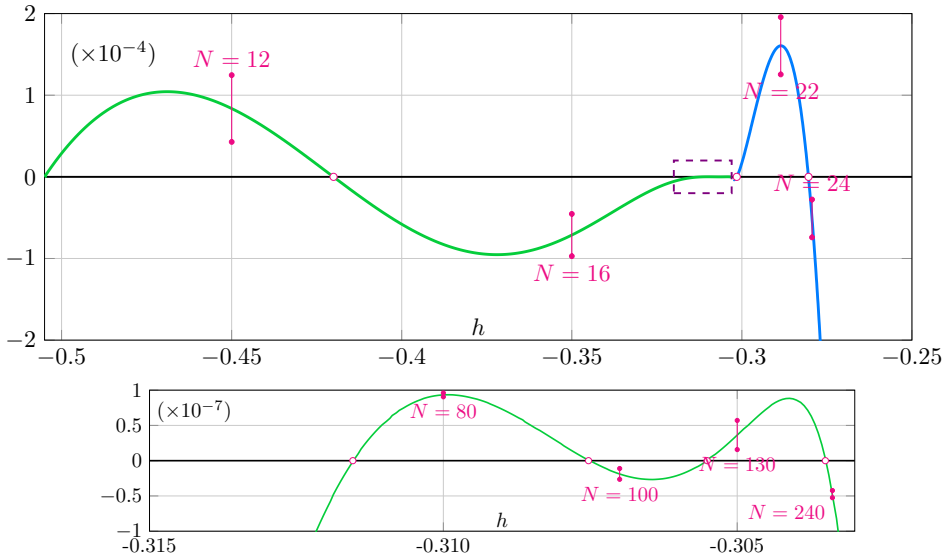
As mentioned above, some mathematicians are certainly reluctant to computer-assisted proofs, and the misadventure related to **[126]** partially justifies this disinclination. Fortunately, we are pleased to announce that we are currently completing the *certified* computations of Abelian integrals using our Coq library, which will give a definitive proof of $\mathcal{H}(4) \geqslant 24$.



(a) Example of a limit cycle with the Van der Pol oscillator.



(b) Unperturbed quartic system of T. Johnson.



(c) Rigorous computation of Abelian integrals shows the expected sign alternations (in magenta: degree for the rigorous polynomial approximations and resulting enclosure of the integral).

■ **Figure I.2**: A computer-assisted proof for $\mathcal{H}(4) \geqslant 24$ in Hilbert's 16th problem.

C Reliability, safety and efficiency for real-life applications

Safety-critical engineering refers to industrial applications with potentially human lives at stake or, at least, large amounts of money or serious environmental damages. Typical examples include aeronautics and aerospace industry, computer-aided medical procedures or nuclear technologies. In such situations, intensive testing phases are usually necessary to increase and validate the reliability of new software. Nevertheless, in addition to their far from negligible cost and time, tests do not guarantee the absence of bugs – they may just detect their *presence*. Therefore, the need for rigorous and formal tools rapidly emerged around those topics – as evidenced by several collaborations between industrial and academic actors, such as the NASA Langley Formal Methods Research Program[10].

The methods developed in this thesis pave the way towards the systematic application of rigorous, symbolic or formal tools to such "real-life" problems. In particular, I focused on aerospace applications. This domain – maybe due to the important amount of funding and public interest it receives, or to a somewhat less rigid legislation than for aeronautics – has been subject to various interesting rigorous and/or symbolic-numeric experiments: Taylor models for rigorous guidance in **[158]**; D-finite recurrences for rigorous evaluation of Taylor series expansion of space debris collision probability in **[227]**; nonrigorous multivariate Chebyshev approximations for the propagation of set of debris in **[255]**.

Chapter 7, in collaboration with Paulo Ricardo Arantes Gilz and Clément Gazzino, makes a contribution to this rigorous space mission design, by validating and using spacecraft trajectories for proximity operations, that is, when relative distances are small compared to the Earth's radius. **Example 4** below presents an application: the *spacecraft rendezvous problem*.

Example 4 : The spacecraft rendezvous problem

The spacecraft rendezvous problem consists, for an active spacecraft (e.g., a shuttle), called the *chaser*, to reach a passive target (e.g., a satellite or the ISS[11]), within a given time interval. To do so, the chaser is equipped with thrusters which can be fired to modify its current orbital trajectory. However, the fuel (more precisely: *hypergolic propellant*) needed by these thrusts is only available in limited quantity. When the tank is empty, the spacecraft cannot be controlled anymore (it is "*dead*"), whence the necessity to minimize the fuel consumption during orbital maneuvers.

We consider a linearized model[12] for the dynamics of the relative motion $X = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T \in \mathbb{R}^6$ of a chaser spacecraft to the passive target:

$$\dot{X}(t) = A(t)X(t) + B(t)u(t), \qquad \textbf{(RDV-i)}$$

where $A(t) \in \mathbb{R}^{6\times 6}$, $B(t) \in \mathbb{R}^{p\times 6}$, and $u(t) \in \mathbb{R}^p$, which represents the control term for the thrusts, lies in a suitable function space. The thrusts are usually modeled as *impulsions*, that is an instantaneous velocity increment rather than an acceleration term. This leads to the so-called *linearized impulsive spacecraft rendezvous problem* (see **Figure I.3a**).

[10] `https://shemesh.larc.nasa.gov/fm/`
[11] International Space Station
[12] This is a usual assumption for the final phase of a rendezvous.

4 (a) *Find the dates t_i and velocity increments ΔV_i, defining the control law $u(t)$, to bring the chaser from its given initial state $X(t_0) = X_0 \in \mathbb{R}^6$ to the prescribed final state $X(t_f) = X_f \in \mathbb{R}^6$ (for fixed $t_0 < t_f$), while minimizing the total fuel consumption:*

$$\begin{aligned} & \inf_u \|u\|_1 = \inf_u \int_{t_0}^{t_f} \|u(t)\| \mathrm{d}t, & \\ s.t. \quad & \dot{X}(t) = A(t)X(t) + B(t)u(t), & t \in [t_0, t_f], \\ & X(t_0) = X_0, \quad X(t_f) = X_f, & t_0,\ t_f \text{ fixed.} \end{aligned} \tag{RDV-ii}$$

In the works **[12, 226]**, which I contributed to, an exchange algorithm is proposed to numerically compute the optimal impulsion dates and values. Roughly speaking, **(RDV-ii)** is restated as a *semi-infinite programming* problem, that is, involving a finite number of variables but an infinite number (i.e., a continuum) of constraints. Starting from an infeasible point, the algorithm selects at each iteration the most violated constraint to reduce the infeasibility (see **Figure I.3b**).

Once the control law has been numerically computed, the rigorous space mission design also requires validated spacecraft trajectories, and in particular a validated final state.

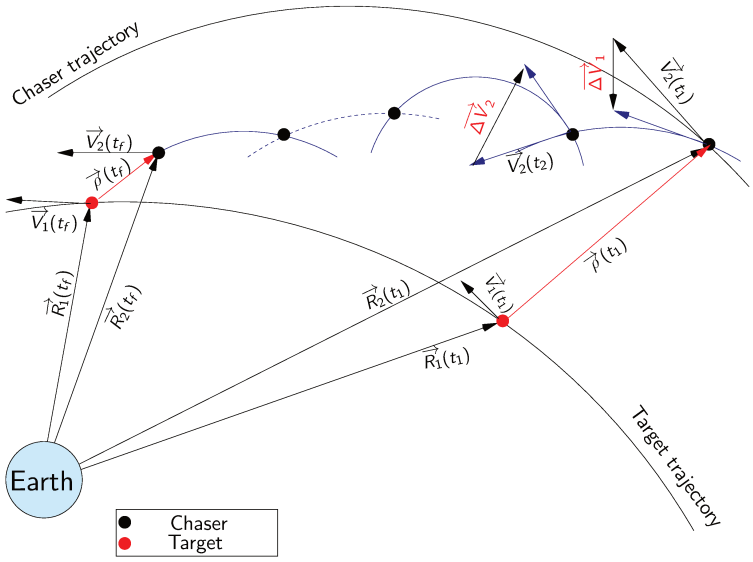
4 (b) *Given the initial state X_0, the impulsion dates t_i and the corresponding velocity increments ΔV_i, compute rigorous enclosures for the actual final state $\widetilde{X_f}$, and check that it is reasonably close to the prescribed X_f.*

Making use of the validation method of **Chapter 5** for coupled systems of LODEs, validated transition matrices using Chebyshev models can be computed for the dynamics **(RDV-i)**, thus allowing for the rigorous propagation of the state $X(t)$ between two consecutive thrusts. In particular, we get the desired enclosure of the final state. **Table I.1** gives the parameters of a concrete ATV mission[13] and the obtained enclosures for the final state in the orbital plane. Obviously, this does not take into account the nonlinearity of the Keplerian dynamics nor the non Keplerian perturbations. However, most control applications related to the spacecraft rendezvous problem consider the linearized dynamics, and being able to certify the numerically computed result in that model is a key feature.

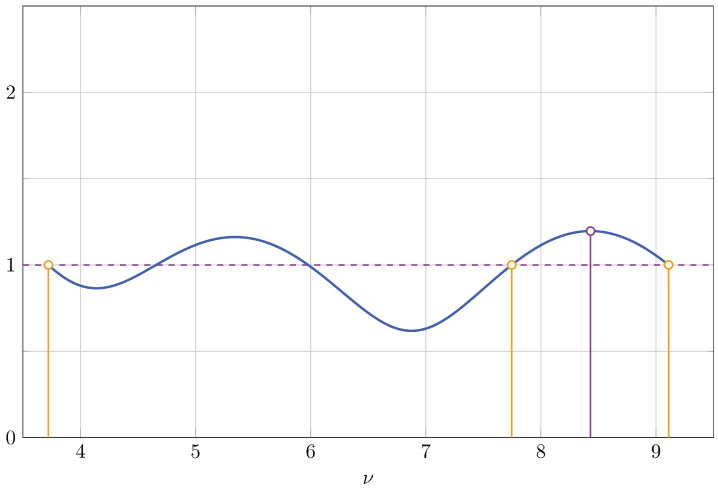
4 (c) *Do I also need to validate the optimality of the control law found in **Question 4 (a)**?*

Well, this particular point is probably not as critical as the final state – one easily imagines the consequences of the ATV supply shuttle entering in collision with the ISS. However, just for the point of being rigorous, one can mention that the algorithm presented in **[12]** provides enclosures for the optimal fuel consumption at each iteration.

[13] *Automated Transfer Vehicle* is the European cargo spacecraft for the supply of the ISS.



(a) The linearized impulsive spacecraft rendezvous problem.



(b) The exchange algorithm for the linearized rendezvous.

■ **Figure I.3**: Numerical solving of the spacecraft rendezvous problem.

Semi-major axis:	6763 km
Eccentricity:	0.0052
Initial time:	$\nu_0 = 0$ rad
Final time:	$\nu_f = 8.1832$ rad
Initial state:	$(x, z, \dot{x}, \dot{z}) =$ $(-30, 0.5, 8.514, 0)$ [km – m/s]
Final state:	$(x, z, \dot{x}, \dot{z}) =$ $(-100, 0, 0, 0)$ [m – m/s]

(a) Given parameters.

ν	$\Delta\dot{x}$ [m/s]	$\Delta\dot{z}$ [m/s]
0.0	-7.50230589	0.742372034
1.388128	-1.55579123	0.08834686
6.666595	0.62565013	0.03325936
8.183058	1.06509710	0.11440204

(b) Computed impulsion dates and velocity increments.

degree	$x(\nu_f)$	$z(\nu_f)$	$\dot{x}(\nu_f)$	$\dot{z}(\nu_f)$
25	-100 + [-2.6861e1, 2.6861e1]	[-7.4084e0, 7.4084e0]	[-1.8133e-2, 1.8133e-2]	[-5.3675e-3, 5.3675e-3]
30	-100 + [-1.0035e-1, 1.0035e-1]	[-2.7676e-2, 2.7676e-2]	[-6.7741e-5, 6.7741e-5]	[-2.0051e-5, 2.0051e-5]
40	-100 + [-2.3194e-5, 2.3190e-5]	[-6.3956e-6, 6.3956e-6]	[-1.5655e-8, 1.5655e-8]	[-4.6336e-9, 4.6336e-9]
50	-100 + [-2.0321e-8, 1.6320e-8]	[-5.0437e-9, 5.0607e-9]	[-1.2358e-11, 1.2376e-11]	[-3.6651e-12, 3.6555e-12]

(c) Validated final state, in function of polynomial approximation degree.

Table I.1: A posteriori validation for an *Automated Transfer Vehicle* (ATV) mission in the orbital plane (x, z).

D Taking into account "lower-level" aspects

The reminder of approximation theory in **Chapter 2** mostly follows a purely mathematical point of view: polynomials are supposed to be given with *real* coefficients, that is, with infinite precision, and the evaluation $x \mapsto p(x)$ is considered exact. Certainly, **Chapters 3**, **4** and **5** rigorously *bound* floating-point errors using interval arithmetic, and they give some insight into how to avoid related overapproximation effects. Nevertheless, no *quantified* analysis of rounding errors is provided.

However, some specific topics require a precise analysis of rounding errors, such as floating-point implementation of functions, where the result's accuracy must be ensured to a given precision (e.g., for correctly rounded functions **[177]**, see **Section 1.2**). In this regard, **Chapter 8** presents a collaboration with Mioara Joldes and Denis Arzelier, in which we propose an exchange algorithm for evaluation *and* approximation error optimized polynomials. As the name suggests, this algorithm has strong connections with the Remez exchange algorithm for best uniform polynomial approximations (see **Section 2.2.2**), and the above-mentioned exchange algorithm for the spacecraft rendezvous **[12]**. Let us take over the example of the Airy function to illustrate our approach.

Example 1 : Airy function Ai – Floating-point implementation

Since Ai cannot be expressed using elementary functions, polynomial approximation is one of the most natural tool to design floating-point implementations of it. It consists, for fixed interval $I = [a, b]$ and degree n, to find the coefficients of a polynomial p in a given basis, so that $\mathrm{Ai}(x)$ will be approximately computed as $p(x)$, for $x \in I$. In an idealistic world without rounding errors, this is the *minimax* problem detailed in **Section 2.2.2**

1 (c) *Given a compact interval $I = [a, b]$ and a degree n, find the best polynomial approximation $p \in \mathbb{R}_n[x]$ for* Ai*, that is the polynomial p minimizing the uniform approximation error:*

$$\operatorname*{arg\,min}_{p \in \mathbb{R}_n[x]} \|\operatorname{Ai} - p\|_{\infty, I} \quad := \quad \operatorname*{arg\,min}_{p \in \mathbb{R}_n[x]} \max_{x \in I} |\operatorname{Ai}(x) - p(x)|.$$

This problem may be solved by the Remez algorithm [209, 208, 202]. In particular, the equioscillating behavior of the resulting approximation error, depicted in **Figure I.4b**, characterizes the optimality.

However, in floating-point (FP) arithmetic, the polynomial p, as a mathematical function $x \mapsto p(x)$, is implemented by a sequence of FP operations, that is a program $\widetilde{p}$, called *evaluation scheme*, that computes a FP result $\widetilde{p}(x)$ on any FP input $x \in I$. In this setting, the coefficients of p are FP numbers, and the result $\widetilde{p}(x)$ is affected by rounding errors. If p^* denotes the optimal polynomial of **Question 1 (c)**, and $\widetilde{p^*}$ an evaluation scheme where the coefficients of p were rounded to FP numbers, then there is no reason for which $\widetilde{p^*}$ would be the optimal solution of the following question (cf. **Figure I.4c**).

1 (d) *Given a compact interval $I = [a, b]$, a degree n, a floating-point precision u and a fixed evaluation scheme $p \mapsto \widetilde{p}$ for degree n polynomials, find a polynomial $p \in \mathbb{R}_n[x]$ with floating-point coefficients that minimizes the total error:*

$$\operatorname*{arg\,min}_{p \in \mathbb{R}_n[x]} \max_{x \text{ FP in } I} |\operatorname{Ai}(x) - \widetilde{p}(x)|.$$

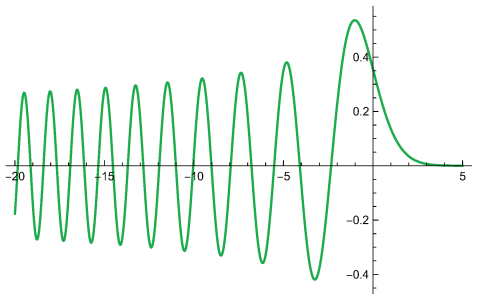
This is an important issue in the domain of computer arithmetic [42, 43]. Our algorithm makes progress on it by optimizing *simultaneously* the approximation and evaluation errors. It consists in a generalization of the Remez algorithm in the framework of semi-infinite programming. The total error of the optimal solution for this Airy example is depicted in **Figure I.4d**, where we can see that the approximation error is "spread" over I, in such a way that the highest peaks are located where the evaluation error is small.

On a final note, this "lower-level" setting is absolutely not incompatible with the rigorous numerics framework discussed in this thesis:

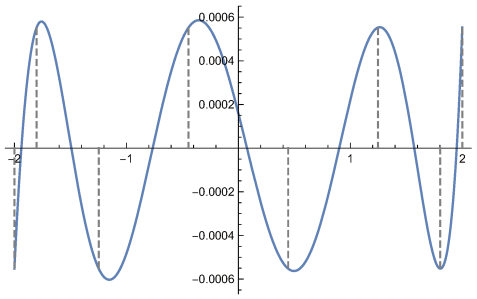
1 (e) *Given a polynomial approximation $p \in \mathbb{R}[x]$ for* Ai*, compute an upper bound over $I = [a, b]$ for the approximation error or the total error.*

For the approximation error, this simply means rigorously bounding $\|\operatorname{Ai} - p\|_{\infty, I}$. Since Ai satisfies a LODE (Ai-ii), a rigorous error bound can be computed using the validation method of **Chapter 4**.

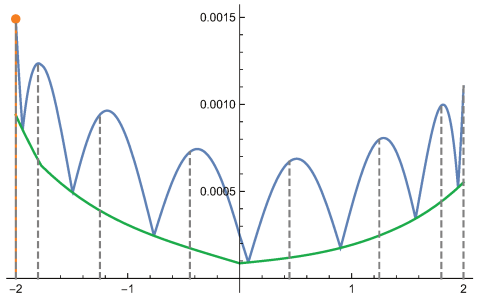
When considering the *total error*, one moreover needs to bound the evaluation error. This can be done using already existing certified software, e.g. Gappa [67] which relies on the previously mentioned Flocq library for Coq.



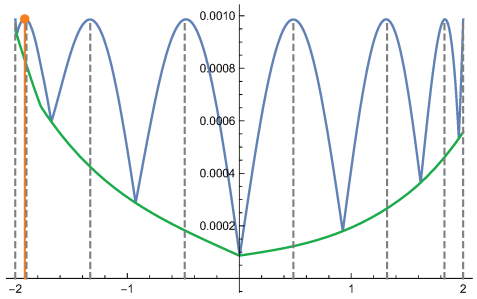
(a) Plot of the Airy function Ai.



(b) Approximation error for the minimax polynomial computed by the Remez algorithm.



(c) Total error for the minimax polynomial p, computed by the Remez algorithm (approx. error in blue and evaluation error in green).



(d) Total error for the best approximation of p found by our method (approx error in blue and evaluation error in green).

■ **Figure I.4**: Polynomial approximation problem for the Airy function Ai, with $I = [-2, 2]$, $n = 6$ and floating-point precision $u = 12$.

E New outlets for rigorous and symbolic-numeric methods

Some domains of mathematics benefited early from the advantages of rigorous numerics, e.g., validated solutions of differential equations, or rigorous/certified optimizers. In contrast, some applications of validated and/or symbolic-numeric methods proposed in this manuscript illustrate new interactions between traditional mathematics and rigorous numerics.

In **Chapter 9**, we (Mioara Joldes, Jean-Bernard Lasserre and I) propose new algorithms to solve inverse problems on measures, that is, reconstructing the support and the density of a measure from its moments. To do so, we make use of the D-finite (and closely related holonomic) setting mentioned above. Although not rigorous, these methods are *symbolic-numeric*: the algebraic machinery of D-finite equations leads to linear systems that are finally solved numerically, since the given moments are usually known to a finite accuracy. Let us exemplify the method with the example below.

Example 5 : Moments of a Gaussian density over a semi-algebraic set

Let $f(x,y) := \exp(a_{xx}x^2 + a_{yy}y^2 + a_{xy}xy + a_x x + a_y y + a_1)$ be an (unnormalized) Gaussian density in the plane, and G the full-dimensional semi-algebraic set of $\mathbb{R}^2$ depicted in **Figure I.5** as the checkered region, whose boundary is given by the zero set of the polynomial:

$$g(x,y) := (x^2+y^2-9)(x^2+y^2-1)((x-2)^2+y^2-1)(x^2+(y-2)^2-1).$$

The moments $m_{\alpha,\beta}$ of the measure $\mu := f\mathbb{1}_G$ (Gaussian measure restricted to G) are defined as:

$$m_{\alpha,\beta} := \int_G x^\alpha y^\beta f(x,y)\mathrm{d}x\mathrm{d}y.$$

Chapter 9 answers the following two questions. First, consider the *direct problem.*

5 (a) *Given the density f and the polynomial boundary g, recover a "complete"[14] set of recurrences for the moments $m_{\alpha,\beta}$, allowing for computing all the moments from a finite number of initial ones.*

This question is partially answered in **Section 9.3** using a heuristic, with more precise statements in the case of exponential-of-polynomial density, in particular Gaussian density. It can be seen as an alternative to usual Creative Telescoping techniques, e.g., **[188]** in the case of semi-algebraic support.

Conversely, here is the *inverse problem.*

5 (b) *Given a finite number of moments $m_{\alpha,\beta}$, try to recover the polynomial in the exponential defining f, and a nontrivial polynomial whose zero set contains the boundary of G.*

Algorithms to solve this problem are given in **Section 9.4**. They consist in "guessing" recurrences for the moments by solving linear systems involving sufficiently many known moments, thus allowing for reconstructing the polynomial boundary and differential equations satisfied

[14] The technical word would be *holonomic* (see **Section 2.1.2**).

by the D-finite density f (see **Figure I.5**). In this exponential-of-polynomial example, the number of needed moments can be a priori bounded.

○ Chebyshev models, widely advocated throughout this thesis, are a powerful tool for problems involving analytic functions over compact intervals. However, such convenient assumptions are not always satisfied: the interval may be open due to singularities, or unbounded for problems with "infinite horizon". Therefore, extending rigorous approximations to nonpolynomial functions (e.g., Bessel or Hermite functions), or even nonlinear families (e.g., rational functions) is a relevant challenge. This is currently an under-examined problem in literature and constitutes one of my near-future research goal. The following question related to **Chapter 6** illustrates the need for such generalized RPAs.

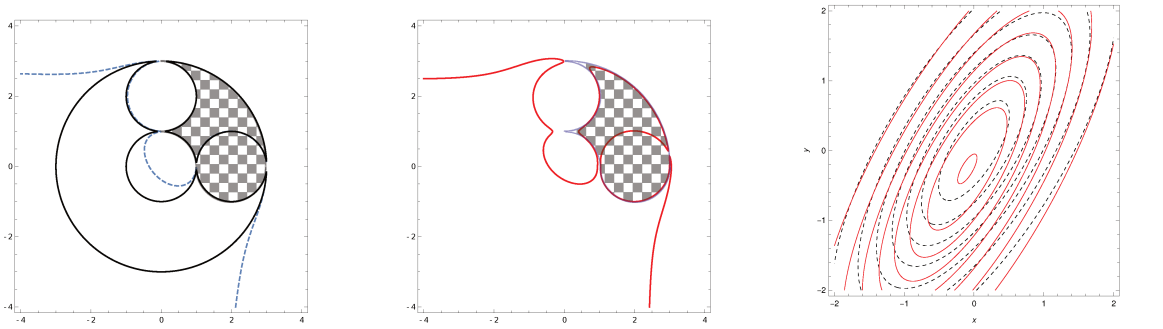
■ **Example 3 : Computing limit cycles in Hilbert's 16th problem**

3 (c) *Is it possible to rigorously prove that T. Johnson's particular example cannot give more than 24 limit cycles?*

To prove that no linear combination of the Abelian integrals under consideration can produce more zeros, we use notions related to Chebyshev systems (defined in **Chapter 2**), in the experimental **Section 6.4**. Specifically, we need to compute the Wronskian of a system of Abelian integrals. Unfortunately, this Wronskian has the following asymptotic at 0, for $t > 0$:

$$W(t) = \sum_{n \geqslant n_1} a_n t^n + \log(t) \sum_{n \geqslant n_2} b_n t^n.$$

with $n_1, n_2 \in \mathbb{Z}$. D-finite techniques and symbolic Laplace transform will hopefully help us in computing (a finite number of) these coefficients a_n, b_n. We also need to develop new rigorous tools to bound the remainder – difficulties arise due to the $\log(t)$ terms and negative powers of t. Hence, this example illustrates a situation where *generalized* RPAs are needed.



(a) Reconstruction (dashed blue) of the algebraic boundary (black) of the support (checkered).

(b) Reconstruction (red) of the support from less accurate moments.

(c) Reconstruction (dashed blue) of a Gaussian density (orange).

■ **Figure I.5**: Example of support and density reconstruction from moments in **Chapter 9**.

F Designing open-source implementations

Besides the theoretical contributions mentioned all along this introduction, I also worked on several implementations. Although not completely mature, they play an important role as a proof of concept for the rigorous and algorithmic approach for function space problems advocated in this thesis, using RPAs and symbolic-numeric tools.

- I developed an open-source C library, named ChebValid[15], that implements the arithmetic on Chebyshev models given in **Chapter 3**, and the validation algorithms for LODEs solutions of **Chapters 4** and **5**. Besides illustrative examples given in these chapters, this library was successfully used in **Chapters 6** and **7** for the rigorous counting of limit cycles (**Example 3**) and the validation of spacecraft trajectories (**Example 4**).
- Assia Mahboubi, Damien Pous and I developed a Coq formalization[16] **[40]** of RPAs with arithmetic operations, with a full implementation on Chebyshev models. More details about it may be found in **Chapter 3**. The Abelian integrals of **Example 3** (cf. **Chapter 6**), that we first computed using ChebValid, have been recomputed with the Chebyshev models of this Coq framework, thus offering the highest confidence level for a computer-assisted proof.
- Mioara Joldes and I also prototyped several scripts in Maple and Mathematica for the problems addressed in **Chapters 8** and **9**. For the former (design of evaluation-error optimized polynomial approximations, see **Example 1**), a complete Mathematica script is available here[17]. For the latter (see **Example 5**), Maple scripts for the different examples in **Chapter 9** may be found here[18].

[15] available at `https://gforge.inria.fr/projects/tchebyapprox/`
[16] available at `http://perso.ens-lyon.fr/florent.brehard/chebapprox/`
[17] available at `http://perso.ens-lyon.fr/fbrehard/EvalMinimax`
[18] available at `http://homepages.laas.fr/fbrehard/HolonomicMomentProblem`

Introduction (en français)

L'anglais, ce n'est jamais que du français mal prononcé.
— Georges Clemenceau

L'analyse numérique est à la base d'un large éventail d'algorithmes traitant des données continues (temps, espace, etc.), largement utilisés, par exemple, dans les simulations en ingénierie, le traitement d'images or la prise de décision automatisée. Le développement d'implémentations extrêmement optimisées et performantes pour de nombreuses routines d'algèbre linéaire, d'analyse fonctionnelle ou d'optimisation continue, est un facteur majeur de cette réussite. Ces implémentations bénéficient grandement de l'efficacité sans cesse accrue des unités de calcul en nombres flottants disponibles dans les processeurs ou autres architectures récentes, sous la houlette du standard IEEE 754.

Néanmoins, un problème très étudié concerne les erreurs numériques inhérentes au calcul scientifique. Habituellement, ces erreurs sont regroupées en deux catégories. La première, intrinsèque au calcul des nombres flottants, est constituée des erreurs d'arrondi, dues à la discrétisation de la droite des réels. La seconde regroupe les erreurs de méthode, lorsque des problèmes de dimension infinie sont approximés en dimension finie, via des méthodes de discrétisation ou de projection. À titre d'exemple, mentionnons les méthodes de Runge-Kutta ou les méthodes spectrales pour résoudre des équations différentielles ordinaires (EDO). Bien que relativement petites en général, ces erreurs s'accumulent au fil de l'exécution du programme et peuvent donner lieu à d'importantes erreurs en fin de compte. Souvent, le problème réside dans le fait que l'on ne peut pas estimer directement la précision du résultat, c'est-à-dire spécifier le nombre de bits corrects, si tant est qu'il y en ait. Par conséquent, certains domaines avec des exigences de sécurité accrues, comme l'ingénierie des systèmes critiques ou les preuves mathématiques assistées par ordinateur, ne peuvent reposer uniquement sur des algorithmes purement numériques.

Objectifs principaux. Au regard de ces insuffisances, nous nous pencherons dans cette thèse sur les défis suivants. Il est à noter que tous n'occuperont pas la même place – le premier recevant en effet le plus d'attention – mais chacun apportera un éclairage propre. Listés ci-dessous, ils seront abordés par la suite dans des sections dédiées, sous l'angle de concepts clés, d'exemples illustratifs et avec pour objectif d'esquisser le plan du manuscrit.

A. Calculer plus rapidement, plus précisément et plus sûrement. Ainsi pourrait-on énoncer l'objectif principal de cette thèse. Après un passage en revue, sans doute per-

sonnel, des techniques existantes en calculs numérique, symbolique et rigoureux (**Chapitres 1** et **2**), nous enrichirons les *approximations polynomiales rigoureuses* (*rigorous polynomial approximations* en anglais, abbrégé en RPA) avec de nouveaux algorithmes, où les méthodes de *validation a posteriori* à base de points fixes jouent un rôle important (**Chapitre 3**). Cela donne lieu à de nouveaux outils efficaces et précis pour des problèmes fonctionnels, notamment les équations différentielles linéaires (**Chapitres 4** et **5**). Qui plus est, nous ne visons pas uniquement le *calcul validé*, mais aussi le *calcul certifié* en recourant à la preuve formelle (**Chapitre 3**).

B. Prouver des théorèmes mathématiques par de nouvelles techniques informatiques. En se basant sur notre travail en cours sur les cycles limites d'un champ de vecteurs polynomial particulier, présenté dans le **Chapitre 6**, nous promouvons l'usage des RPA, de la validation *a posteriori* et des méthodes symboliques-numériques dans l'élaboration d'outils efficaces pour les preuves mathématiques assistées par ordinateur. Tirant parti de notre approche *a posteriori*, ou à base de *certificats*, il devient plus facile d'introduire la preuve formelle dans le processus de preuve par ordinateur, offrant ainsi le plus haut niveau de confiance possible.

C. Sécurité et efficacité pour des applications réelles. Pouvoir calculer de manière plus fiable est crucial pour des applications relevant de l'ingénierie critique. Pour autant, nous ne pouvons pas (trop) sacrifier les exigences d'efficacité. En calculant et utilisant des trajectoires de satellites validées lors d'opérations spatiales de proximité (**Chapitre 7**), nous illustrons comment nos méthodes peuvent s'appliquer dans un contexte critique, comme la conception rigoureuse de missions spatiales.

D. Intégrer les aspects « bas niveau » est essentiel lors l'élaboration d'outils et d'implémentations efficaces pour des applications « dans la vraie vie ». Cet aspect est également pris en compte dans cette thèse, bien que dans une moindre mesure. À titre d'exemple, le **Chapitre 8** présente une contributions dans ce sens, à savoir un nouvel algorithme d'échange pour calculer des polynômes d'approximation avec erreur d'évaluation optimisée. Cela se révèle utile pour l'implémentation numérique de fonctions mathématiques, puisque la précision flottante sous-jacente est finie.

E. Nouveaux champs d'application des méthodes rigoureuses et symboliques-numériques. Introduire des techniques symboliques-numériques et de calcul rigoureux dans d'autres domaines des mathématiques constitue un défi particulièrement intéressant, permettant en retour d'enrichir nos outils avec de nouveaux concepts mathématiques. Par exemple, la reconstruction à partir des moments du support et de la densité d'une mesure, présentée dans le **Chapitre 9**, utilise les *polynômes de Ore* – un outil symbolique – dans le cadre d'un problème inverse.

F. Concevoir des implémentations *open-source*. Dernièrement mais non des moindres, La production de code logiciel libre est une contribution essentielle de cette thèse, qui vient corroborer notre *manifesto* : fournir des méthodes algorithmiques et efficaces. Par ailleurs, ces implémentations furent d'une grande aide pour les applications sus-mentionnées.

A Calculer plus rapidement, plus précisément et plus sûrement

Les cinq premiers chapitres de ce manuscrit visent cet objectif. Pour commencer, voici les trois *niveaux de sécurité* du calcul dont il sera question :

1. Une première couche *purement numérique* regroupe les algorithmes d'analyse numérique classique, avec en ligne de mire la précision et l'efficacité. Bien que souvent accompagnés de résultats de convergence asymptotique, ces algorithmes ne peuvent en général pas garantir la précision du résultat sur une instance donnée.

2. Le deuxième niveau est celui des algorithmes *validés* au sens large, dans le sens où le résultat calculé satisfait des propriétés mathématiques précises. Nous distinguerons notamment :

   - le *calcul symbolique* (**Objectif A.1**) pour des problèmes essentiellement de nature algébrique, travaillant avec des *représentations exactes* des objets et parfois accompagnés de *certificats* rendant possible une vérification a posteriori du résultat.
   - le *calcul rigoureux* (**Objectif A.2**) reposant sur des représentations *ensemblistes* numériques, et fournissant ainsi des encadrements rigoureux pour le résultat exact du problème, qui peut ne pas être représentable exactement en machine.

   Parfois, calculs symbolique et numérique (rigoureux) peuvent être combinés dans des méthodes dites *symboliques-numériques*.

3. Une troisième couche, celle du *calcul certifié*, offre une sécurité maximale, à savoir au niveau de l'implémentation. La *preuve formelle* (**Objectif A.3**) ouvre la voie à des implémentations certifiées, dont la correction est garantie par un logiciel appelé *assistant de preuve*.

Le choix du niveau de sécurité adéquat dépend fortement de l'application visée : sommes-nous prêts à dédier plus de temps à un calcul donné pour augmenter la confiance que nous avons dans le résultat ? Toutefois, ce choix est également limité par les outils et logiciels disponibles. En particulier, les implémentations certifiées de méthodes numériques sont encore rares et restent souvent bien moins efficaces que leurs analogues non rigoureux.

Avant d'aborder les contributions de cette thèse, un aperçu des outils de calculs nécessaires est donné dans les deux premiers chapitres. Le **Chapitre 1** s'intéresse aux représentations des nombres : représentations exactes pour le calcul symbolique, les nombres flottants (très efficaces mais sujets aux erreurs d'arrondis), et enfin les intervalles pour encadrer rigoureusement les nombres réels. Ensuite, le **Chapitre 2** aborde la représentation des fonctions en insistant sur les deux sujets suivants : le cadre symbolique des fonctions D-finies, qui sont les solutions des équations différentielles ordinaires linéaires (EDOL) à coefficients polynomiaux, ainsi qu'un survol de certains résultats de théorie de l'approximation polynomiale qui se révélera précieux pour les chapitres à suivre.

Les approximations polynomiales rigoureuses (RPA), qui occupent une place centrale dans cette thèse, seront présentées dans le **Chapitre 3**, en même temps que seront introduites les *méthodes de validation a posteriori* à base de *point fixes* qui seront utilisées pour définir la

division et la racine carrée de RPA. Il en résulte une arithmétique des RPA qui sert de fondement à des méthodes rigoureuses plus élaborées pour des problèmes d'analyse fonctionnelle. En particulier, une contribution centrale de cette thèse consiste en des algorithmes calculant des RPA dans la base de Tchebychev pour les solutions d'EDOL, conçus avec Nicolas Brisebarre et Mioara Joldes. Ils seront présentés dans les **Chapitres 4** (pour les EDOL scalaires) et **5** (pour les systèmes d'EDOL couplées).

L'exemple suivant autour de la fonction d'Airy Ai, servira à illustrer comment ces différents outils détaillés dans ce qui suit apportent des points de vue complémentaires sur des mêmes questions.

Example 1 : Fonction d'Airy Ai

La fonction d'Airy Ai (cf. **Figure I.9a**) est une fonction spéciale fréquemment rencontrée en mathématiques **[2]**, qui peut être définie via l'intégrale de Riemann impropre suivante :

$$\mathrm{Ai}(x) = \frac{1}{\pi}\int_0^{+\infty} \cos\left(\frac{t^3}{3} + xt\right) \mathrm{d}t, \tag{Ai-i}$$

ou par l'équation différentielle du second ordre suivante avec conditions initiales en 0, où Γ désigne la fonction Gamma :

$$\mathrm{Ai}''(x) - x\,\mathrm{Ai}(x) = 0, \qquad \mathrm{Ai}(0) = \frac{1}{3^{\frac{2}{3}}\Gamma\left(\frac{2}{3}\right)}, \qquad \mathrm{Ai}'(0) = -\frac{1}{3^{\frac{1}{3}}\Gamma\left(\frac{1}{3}\right)}. \tag{Ai-ii}$$

Cette fonction n'admet pas de forme close plus simple à base de fonctions élémentaires, comme exp, sin, cos ou leurs réciproques. Un des objectifs de l'analyse numérique, de calcul formel et du calcul rigoureux consiste à développer des méthodes pour manipuler de telles fonctions, notamment l'évaluer en un point, vérifier des égalités, produire des approximations validées, etc.

A.1 Calcul symbolique

Les méthodes de calcul symbolique **[257]**, qui reflètent notre manière usuelle de faire des mathématiques en passant par des *représentations exactes* des nombres, fonctions, etc., constituent un moyen naturel de contourner les problèmes d'erreurs numériques. Plusieurs décennies de recherche dans ce domaine ont abouti à toute une collection d'algorithmes efficaces, bien étudiés en terme de complexité et implémentés dans des logiciels de calcul formel faciles à prendre en main, tels Maple ou Mathematica. Les *fonctions D-finies*, présentées dans ce qui suit en raison de leurs multiples apparitions tout au long de cette thèse, forment un exemple remarquable d'objets avec représentations exactes en machine.

Fonctions D-finies. Ces fonctions, qui sont les solutions des EDO linéaires à coefficients polynomiaux **[234]**, sont omniprésentes en mathématiques – elles représentent environ 60% des fonctions répertoriées dans **[2]**, dont la fonction d'Airy Ai donnée dans l'**Exemple 1**. Grâce aux nombreux efforts d'une communauté active autour de ce sujet, des logiciels de calcul formel comme Maple ou Mathematica permettent aujourd'hui de manipuler de telles fonctions comme des *structures de données* (une équation différentielle annulant la fonction,

avec suffisamment de conditions initiales) et d'agir sur elles via des opérations arithmétiques et différentielles.

Example 1 : Fonction d'Airy Ai – un peu de calcul formel

Nous illustrons ici comment l'algorithmique des fonctions D-finies permet de prouver des résultats non triviaux.

1 (a) *Prouvons l'identité suivante* **[2,** Eq. (10.4.15)**]** *:*

$$\mathrm{Ai}(-x) = \sqrt{\frac{x}{9}}\left(J_{\frac{1}{3}}\left(\frac{2}{3}x^{\frac{3}{2}}\right) + J_{-\frac{1}{3}}\left(\frac{2}{3}x^{\frac{3}{2}}\right)\right), \qquad x > 0, \tag{Ai-iii}$$

où les fonctions de Bessel $J_{\pm\frac{1}{3}}$ *forment une base des solutions de :*

$$9z^2 f''(z) + 9z f'(z) + (9z^2 - 1)f(z) = 0. \tag{$J_{\frac{1}{3}}$}$$

En utilisant les opérations de clôture implémentées sous forme l'*algorithmes*, le paquet GFUN[19] pour MAPLE *calcule* automatiquement que le membre droit dans (Ai-iii) vérifie $g''(x)+xg(x) = 0$. En effet, cette expression se déduit de $J_{\pm\frac{1}{3}}$ par une substitution algébrique $z \mapsto \frac{2}{3}x^{\frac{3}{2}}$, suivie d'une somme, et finalement multipliée par $x \mapsto \sqrt{\frac{x}{9}}$. Or $x \mapsto \mathrm{Ai}(-x)$, de par (Ai-ii), vérifie la même EDOL, de sorte que vérifier l'égalité des conditions initiales en 0 suffit pour établir l'identité (Ai-iii), par le théorème de Cauchy-Lipschitz.

Télescopage Créatif. Le Télescopage Créatif désigne un ensemble de techniques **[269, 59, 141]** développées dans le but de calculer automatiquement des équations différentielles ou de récurrence pour des intégrales ou des sommes de quantités D-finies, permettant même parfois de retrouver des formes closes non triviales. Qui plus est, ces techniques plutôt complexes produisent également un certificat permettant de revérifier le résultat *a posteriori* (voir **Chapitre 2** pour plus de détails).

Example 2 : Fonction génératrice exponentielle des polynômes de Tchebychev

2 (a) *Prouvons l'identité suivante :*

$$\sum_{n=0}^{+\infty} T_n(x)\frac{t^n}{n!} = e^{tx}\cosh(t\sqrt{x^2-1}), \qquad |x| < 1,$$

où T_n *désigne le n-ième polynôme de Tchebychev, défini dans le Chapitre 2.*

L'intégrande $f(t,n) := T_n(x)\frac{t^n}{n!}$ vérifie les équations mixtes différentielles/aux différences :

$$t\partial_t \cdot f - nf = 0,$$
$$(2+3n+n^2)S_n^2 \cdot f + (-2tx - 2ntx)S_n \cdot f + t^2 f = 0.$$

avec $\partial_t \cdot f(t,n) := \frac{\partial f}{\partial t}(t,n)$ et $S_n \cdot f(t,n) := f(t,n+1)$.

[19] `http://perso.ens-lyon.fr/bruno.salvy/software/the-gfun-package/`

En utilisant par exemple la routine `CreativeTelescoping` du paquet HolonomicFunctions[20] de Mathematica, on obtient l'équation différentielle suivante pour $g(t) := \sum_{n=0}^{+\infty} f(t,n)$:

$$\partial_t^2 \cdot g - 2x\partial_t \cdot g + g = 0.$$

Pour finir, les procédures usuelles de résolution d'équations différentielles de Maple ou Mathematica permettent de retrouver l'expression en forme close $e^{tx}\cosh(t\sqrt{x^2-1})$ à partir de cette équation différentielle.

Limites des méthodes symboliques. Cependant, certains problèmes sont par essence insolubles par des tels outils, notamment pour les raisons suivantes :

- Seules certaines classes restreintes de nombres ou fonctions admettent des représentations exactes, par exemple les nombres ou fonctions rationnels, algébriques, etc. Considérons cependant à titre d'exemple la constante d'Euler $\gamma := \lim_{n\to\infty}\left(\sum_{k=1}^{n}\frac{1}{k} - \log n\right)$. Quel comportement calculatoire attendons-nous des algorithmes sur celui-ci ? – alors même que savoir si γ est rationnel ou pas reste un problème ouvert.
- Certains objets ont des représentations exactes mais implicites. Ainsi, au moment de tester des propriétés concrètes, une étape de calcul rigoureux peut s'avérer nécessaire pour convertir la représentation implicite en des valeurs numériques. C'est par exemple le case pour les fonctions D-finies, pour lesquelles des approximations polynomiales rigoureuses seront construites dans les **Chapitres 4** et **5** (voir ci-dessous).
- Même lorsqu'il est possible de travailler avec des représentations exactes, leur taille intrinsèque peut se révéler être un obstacle majeur pour une implémentation en machine. Cela arrive souvent pour les algorithmes où l'on itère un grand nombre de fois des opérations algébriques sur des objets représentés exactement. Dans le **Chapitre 1**, l'**Exemple 1.3** illustrera ce phénomène avec l'inversion de matrices à coefficients rationnels.

A.2 Calcul rigoureux et méthodes de validation

Le calcul rigoureux **[247]** s'affranchit des limitations des méthodes symboliques en substituant aux représentations exactes des représentations *validées* dans divers problèmes, notamment en analyse fonctionnelle, par exemple les équations différentielles ou le contrôle optimal. L'idée principale est de calculer des représentations ensemblistes effectives, comme des intervalles réels ou des boules dans des espaces fonctionnels, avec la garantie que la solution exacte du problème en question soit contenue dans cet ensemble de valeurs possibles. Cette définition, plus souple que les représentations exactes, permet de recourir aux nombres flottants et aux algorithmes numériques, tout en restant correct grâce à un usage attentif des modes d'arrondis et des bornes d'erreur *a posteriori*.

Arithmétique des intervalles. C'est une composante de base essentielle du calcul rigoureux (cf. **[173, 247, 221]**), qui sera présentée dans le **Chapitre 1**. L'idée consiste à utiliser

[20] `https://www3.risc.jku.at/research/combinat/software/ergosum/RISC/HolonomicFunctions.html`

des intervalles réels avec bornes représentables (en général, des nombres flottants) pour encadrer rigoureusement des réels, et à fournir des opérations qui préservent cette correction. Par exemple, de $\pi \in [3.1415, 3.1416]$ et $e \in [2.7182, 2.7183]$, nous déduisons :

$$\pi + e \in [3.1415, 3.1416] \oplus [2.7182, 2.7183] = [5.8597, 5.8599].$$

Par conséquent, le fait de remplacer toutes les opérations de nombres flottants par leurs analogues en intervalles résout en principe le problème des erreurs d'arrondis. Néanmoins, certaines limitations, détaillées dans le **Chapitre 1**, nous amènent à modérer ce point de vue faisant de l'arithmétique des intervalles l'alpha et l'oméga du calcul rigoureux :

- Les surestimations constituent une faiblesse bien connue de l'arithmétique d'intervalles **[221]**, en raison de divers phénomènes celui de la *dépendance* ou du *wrapping effect* : l'intervalle calculé est correct, mais trop grand pour donner quelque information utile.

- Bien que l'intervalle calculé soit correct par rapport aux erreurs d'arrondi, les erreurs de méthodes ne sont pas prises en compte, étant donné qu'elles résultent de la discrétisation du problème lui-même, et non simplement de l'approximation des réels par des nombres flottants.

Approximations polynomiales rigoureuses (RPA). Les RPA ont été conçus comme analogues en dimension supérieure à l'analyse d'intervalles. Ce sont simplement des polynômes assortis d'une borne supérieure de l'erreur entre ce polynôme et la fonction représentée, selon une norme donnée (cf. **Figure I.6**). Ces idées furent initialement conceptualisées dans les années 1980 avec l'*ultra-arithmétique* **[79, 80]**. Quelque temps plus tard, l'implémentation des *modèles de Taylor* **[164, 165]** (que l'on peut voir comme des séries de Taylor tronquées rigoureusement) dans la bibliothèque COSY Infinity raviva l'intérêt porté aux RPA.

Cependant, le **Chapitre 2** souligne les limites des développements de Taylor en théorie de l'approximation, là où les *séries de Fourier généralisées* – et en particulier les séries de développements en séries de Tchebychev – sont souvent bien plus efficaces. Partant de ce constat, les *modèles de Tchebychev*, conçus dans **[44, 129]**, furent rapidement adoptés pour certaines preuves assistées par ordinateur (voir par exemple **[153]**). Plus encore, l'approche pleinement algorithmique des modèles de Tchebychev (par exemple dans **[19]**) a constitué une source d'inspiration majeure pour cette thèse.

Le **Chapitre 3** présente une arithmétique pour les modèles de Tchebychev, avec quelques différences par rapport à **[129]**, notamment pour la division et la racine carrée qui sont ici effectuées selon une approche interpolation – validation *a posteriori*. Nous y présenterons également une bibliothèque C nommée ChebValid[21] pour les modèles de Tchebychev, ainsi qu'une formalisation des RPA en Coq[22] (cf. **Objectif F**). Le **Chapitre 4** repose également sur les *méthodes de validation a posteriori par point fixe* (voir ci-dessous) pour calculer de manière automatisée des RPA pour des solutions d'EDOL, dont les coefficients sont eux-mêmes représentés par des RPA. Contrairement à **[19]**, qui se limite aux fonctions D-finies, notre approche se rapproche davantage dans l'esprit des méthodes de validation de type Newton, présentées dans ce qui suit.

[21] disponible sur `https://gforge.inria.fr/projects/tchebyapprox/`
[22] disponible sur `http://perso.ens-lyon.fr/florent.brehard/chebapprox/`

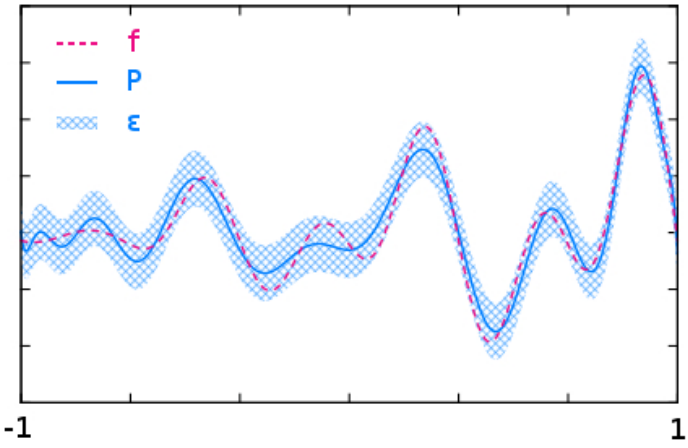


Figure I.6 : Un RPA pour le problème du rendez-vous spatial : une approximation polynomiale + une borne d'erreur rigoureuse (en bleu) forment un tube contenant la trajectoire exacte (en pointillés magenta).

Validation *a posteriori* par point fixe. Ces méthodes constituent un outil puissant pour le calcul rigoureux lorsque des techniques *directes* (aussi appelées *auto-validantes*) ne sont pas applicables ou insuffisamment précises. Le problème se résout alors en deux étapes totalement indépendantes :

1. L'étape d'*approximation* calcule une approximation x° de la solution exacte x^* du problème en question. N'importe quel algorithme numérique peut ici être utilisé – aucune hypothèse n'est requise.

2. L'étape de *validation* reformule le problème initial de sorte à faire de x^* l'unique point fixe d'un opérateur contractant bien choisi, puis reconstruit *a posteriori* une borne rigoureuse sur l'erreur de x° par rapport à x^*, en utilisant le *théorème du point fixe de Banach*.

Cette approche en deux étapes, que nous présenterons plus en détails dans le **Chapitre 3**, est souvent utilisée dans les preuves assistées par ordinateur pour des problèmes d'analyse fonctionnelle et de systèmes dynamiques, notamment via les *méthodes de validation a posteriori de type Newton* (voir par exemple **[137, 136, 178, 199, 189, 265, 9, 250, 153, 111]**, ainsi que **[179]** et les références qui y sont citées). Allant à l'encontre de la stratégie dominante du *cas par cas* de ces travaux, la méthode de validation pour les solutions d'EDOL présentée dans le **Chapitre 4** suit une approche totalement algorithmique avec une analyse de complexité détaillée, et a de plus été implémentée dans la bibliothèque C ChebValid mentionnée plus haut.

Toujours en rapport avec la validation *a posteriori*, le **Chapitre 5** présente une nouvelle généralisation du théorème du point fixe de Banach, donnant un cadre général pour la validation fine composante par composante de problèmes vectoriels. Ceci s'applique à la validation de systèmes d'EDOL couplées, en réutilisant les techniques du **Chapitre 4**.

Example 1 : Fonction d'Airy Ai **– approximation rigoureuse**

1 (b) *Comment calculer efficacement des RPA précis pour* Ai *?*

La fonction d'Airy Ai abordée précédemment dans le cadre du calcul symbolique, peut maintenant être rigoureusement approchée et évaluée à l'aide de modèles de Tchebychev. La **Section 4.5.1** explique en détails comment la méthode de validation pour EDOL calcule et valide des approximations de Ai. Il est important de noter que cette méthode de validation peut s'appliquer sur n'importe quelle approximation polynomiale, pas uniquement les séries de Tchebychev tronquées du **Chapitre 4** : par exemple, les polynômes optimisés pour l'erreur d'évaluation calculés dans le **Chapitre 8** (cf. **Objectif D**).

A.3 Preuve formelle et implémentations certifiées

Un niveau de confiance encore plus élevé pour le calcul rigoureux peut-être souhaitable dans certains cas, c'est-à-dire des implémentations *certifiées* grâce à l'utilisation de la preuve formelle. Un *assistant de preuve* est un logiciel dans lequel le programmeur édite des scripts de preuves à la fois pour des algorithmes ou des énoncés mathématiques. Ces scripts de preuves sont ensuite vérifiés ligne par ligne par le noyau, qui consiste en un nombre relativement restreint de lignes de code sur lequel repose toute la correction mathématique de ce procédé de vérification. Voici les noms de quelques uns des plus célèbres assistants de preuve : Mizar, ACL2, HOL4, HOL-Light, Isabelle, Coq, PVS, Lean, etc. Bien que plusieurs d'entre eux proposent des outils d'automatisation (comme les tactiques en Coq ou l'outil Sledgehammer en Isabelle), prouver formellement des programmes est une tâche qui requiert souvent beaucoup de temps et de main d'œuvre. En retour, certaines formalisations d'importants théorèmes ou conjectures ont suscité une reconnaissance bien mérités et ont ainsi contribué à renforcer la place des preuves assistées par ordinateur au sein de la communauté mathématique (voir **Objectif B** plus bas).

Dans ce qui suit, je ciblerai plus particulièrement l'assistant de preuve Coq[23] **[22]** développé à l'INRIA, qui est celui utilisé dans notre formalisation des RPA (voir **Objectif F**). Voici quelques notions élémentaires sur la *théorie des types* à la base de la logique de Coq.

Théorie des types et preuves. Les langages de programmation dits *typés* classent les données dans différentes catégories (prédéfinies ou créées par l'utilisateur), que l'on appelle *types*. La notation `a : A` signifie : « le terme `a` a pour type `A` ». Par exemple, en Coq, `nat` représente les entiers naturels, `bool` les booléens, `A → B` les fonctions prenant un argument de type `A` et renvoyant un résultat de type `B`, etc. La *correspondance de Curry-Howard* **[92,** Chap. 3**]** permet d'interpréter les types comme des propositions mathématiques, et les termes de ce type comme des preuves de la proposition correspondante. Ainsi par exemple, une preuve de l'implication mathématique $A \Rightarrow B$ n'est rien d'autre qu'une fonction de type `A → B`, qui construit une preuve de B à partir d'une preuve de A.

Le langage de preuve formelle Coq enrichit cette correspondance grâce aux *types dépendants* **[169][22,** Chap. 4**]**. Un type dépendant est tout simplement un type $B(a)$ paramétré par des habitants a du type A, donc une terme `B : A → Type`. Le *produit dépendant* `∀ (a : A), B a` généralise le type flèche classique : il contient les fonctions f qui associent à chaque argument `a : A` une valeur de retour `f a : B a`. Il est donc clair que le produit dépendant est l'analogue du

[23] `https://coq.inria.fr/`

quantificateur universel en logique mathématique (et de manière similaire, il existe des *sommes dépendantes* pour que quantificateur existentiel). Pour finir, Coq permet également à l'utilisateur de définir des *types inductifs* **[22,** Chap. 6**]**, dont les éléments sont construits à partir d'un nombre fini de constructeurs déclarés par le programmeur. Par exemple, les entiers naturels standards de Coq sont donnés par le type inductif suivant, qui correspond à une représentation unaire des entiers (cf. **Chapitre 1**) :

```
Inductive nat : Set :=
    | O : nat
    | S : nat → nat.
```

Coq associe automatiquement un *principe d'induction* à cette définition, qui en gros stipule : « Tous les motifs possibles construit avec `O` et `S`, c'est-à-dire `S (S (... (S O)))`, sont des termes deux à deux distincts de `nat`, et tous les termes de `nat` sont de cette forme ».

```
nat_rect : ∀ P : nat → Type, P O → (∀ n : nat, P n → P (S n)) → ∀ n : nat, P n.
```

Nous terminons ici cette courte introduction à la théorie des types, et renvoyons le lecteur intéressé à **[92, 22]** pour plus de détails. En somme, la théorie des types peut servir de fondations logiques aux mathématiques, à la place de la théorie des ensembles. Dans cette optique, le programme *Fondations univalentes*[24] entend redéfinir toutes les mathématiques via une extension nommé *Théorie homotopique des types* (*Homotopy Type Theory* en anglais, HoTT) **[204]**. À présent, nous nous intéressons plus particulièrement à la formalisation de l'analyse mathématique dans Coq.

Analyse en Coq. Voilà un sujet largement largement discuté au sein de la communauté de la preuve formelle. Une des raisons principales est que dans la logique purement constructive de Coq, les différentes constructions usuelles du corps des nombres réels (coupures de Dedekind, suites de Cauchy, etc.) ne sont pas équivalentes. Qui plus est, une définition purement constructive des nombres réels implique forcément que certaines propriétés importantes ne seront pas satisfaites, comme le test à zéro `∀ (x : R), x = 0 \/ x <> 0`. Ainsi, la bibliothèque standard de Coq fait le choix d'un type axiomatisé `R` pour les nombres réels, bien que des constructions alternatives puissent également être utilisées, comme la bibliothèque C-CoRN/MathClasses **[63]**.

Basée sur l'axiomatisation des réels proposée dans la bibliothèque standard, la bibliothèque Coquelicot **[29]** offre un cadre agréable pour l'analyse réelle et fonctionnelle. Notre formalisation du théorème du point fixe de Banach, que nous utilisons pour les certaines opérations sur les RPA, repose largement sur les filtres par lesquels Coquelicot définit les notions de topologie (voir **Chapitre 3**).

Du côté de l'analyse numérique, la bibliothèque Flocq **[30]** formalise l'arithmétique des nombres flottants, qui à sont tour est utilisée par l'arithmétique d'intervalles, formalisée dans la bibliothèque CoqInterval **[171]**. Cette dernière offre la possibilité de prouver automatiquement des inégalités réelles par le biais de l'arithmétique d'intervalles. Plus proche encore de nos travaux, la bibliothèque CoqApprox **[168]** formalise les modèles de Taylor, augmentant ainsi les possibilités offertes par CoqInterval.

B Prouver des théorèmes mathématiques par de nouvelles techniques informatiques

[24] `https://homotopytypetheory.org/`

Le principe de preuve assistée par ordinateur consiste à déléguer à la machine tout ou partie d'une preuve d'un théorème mathématique. Plusieurs raisons peuvent conduire à ce choix quelque peu audacieux : des calculs trop compliqués pour être faits à la main, une explosion combinatoire des cas à analyser, un très haut niveau d'abstraction pouvant rendre la preuve douteuse, etc. L'expérience personnelle de V. Voevodsky, qu'il relate dans sa note « The Origins and Motivations of Univalent Foundations »[25], insiste sur les limites des mathématiques usuelles sur papier :

« *I can see two factors that contributed to this outrageous situation : Simpson claimed to have constructed a counterexample, but he was not able to show where the mistake was in our paper. Because of this, it was not clear whether we made a mistake somewhere in our paper or he made a mistake somewhere in his counterexample. Mathematical research currently relies on a complex system of mutual trust based on reputations. By the time Simpson's paper appeared, both Kapranov and I had strong reputations. Simpson's paper created doubts in our result, which led to it being unused by other researchers, but no one came forward and challenged us on it.* »

Les preuves assistées par ordinateur en analyse mathématique font souvent appel aux nombres flottants et aux algorithmes numériques. Toutefois, elles doivent garantir des propriétés mathématiques sur les quantités calculées – un théorème ne peut pas être correct *modulo* des erreurs numériques ! Dans cette optique, le calcul rigoureux avec représentations ensemblistes décrit dans l'**Objectif A.2** est un ingrédient essentiel qui a joué un rôle majeur dans plusieurs preuves célèbres, comme la preuve par Landford de la conjecture de Feigenbaum **[147]**, la preuve par Tucker sur l'existence de l'attracteur de Lorenz **[246]**, ou la preuve par Hales de la conjecture de Kepler **[102]**. Des bibliothèques de calcul rigoureux ont également été développées spécialement pour les preuves assistées par ordinateur, comme la bibliothèque CAPD[26] pour la théorie des systèmes dynamiques.

Il est clair que déléguer certaines parties d'une preuve à la machine est une décision tout sauf anodine, de sorte que les preuves assistées par ordinateurs peuvent être parfois mal reçues par certains mathématiciens, qui pourraient douter de la rigueur de ces approches, ou simplement refuser de croire en un résultat qu'ils ne pourraient pas vérifier à la main, ligne par ligne. Pour cette première objection, [27], le recours aux *assistants de preuve formelle* devrait définitivement clore ce débat, comme ce fut par exemple le cas pour le théorème des quatre couleurs **[95]** par Gonthier *et al.* en Coq, le projet Flyspeck **[101]** par Hales *et al.* en HOL-Light pour la conjecture de Kepler, ou encore la formalisation par Immler **[119]** en Isabelle de la preuve assistée par ordinateur de Tucker sur l'existence de l'attracteur de Lorenz.

Le travail mené conjointement avec Nicolas Brisebarre, Mioara Joldes et Warwick Tucker, et présenté dans le **Chapitre 6**, est un exemple de preuve assistée par ordinateur en analyse, utilisant dans un premier temps des calculs rigoureux, puis certifiés par preuve formelle en un second temps. Ainsi, nous avons utilisé à la fois la bibliothèque C ChebValid C et le développement Coq (cf. **Objectif F**), dans le but de prouver ce qui suit.

Example 3 : Calculer des cycles limites dans le cadre du 16ème problème de Hilbert

[25] `https://www.ias.edu/ideas/2014/voevodsky-origins`
[26] `http://capd.ii.uj.edu.pl/`
[27] Pour la seconde en revanche, c'est essentiellement une question de goût personnel – et, après tout, vous n'êtes pas obligé d'aimer une preuve pour l'accepter.

Dans le **Chapitre 6** de ce manuscrit, nous nous intéressons au calcul des cycles limites dans le cadre du 16ème problème de Hilbert, qui s'énonce ainsi :

« *Quel est le nombre maximal $\mathcal{H}(n)$ de cycles limites qu'un champ de vecteurs polynomial dans le plan de degré au plus n peut avoir, c'est-à-dire :*

$$\begin{cases} \dot{x} = P(x, y) \\ \dot{y} = Q(x, y) \end{cases}$$

où $P, Q \in \mathbb{R}[x, y]$ sont de degré au plus n ? »

Plus de précisions et définitions seront données dans le chapitre concerné. Rapidement, un *cycle limite* est une orbite périodique isolée, attirant ainsi localement les trajectoires à proximité (en temps positif ou négatif). L'oscillateur de Van der Pol, représenté sur la **Figure I.7a**, constitue un exemple célèbre de cycle limite.

Nous n'attaquerons pas le 16ème problème de Hilbert en soi : nous nous contenterons de construire un exemple afin de calculer une nouvelle *borne inférieure* pour $\mathcal{H}(4)$. En réalité, notre point de départ fut l'article **[126]**, dans lequel T. Johnson prétend isoler rigoureusement 26 cycles limites pour un système quartique construit à cet effet. La preuve, basée sur le théorème de Poincaré-Pontryagin qui relie les cycles limites aux zéros d'*intégrales abéliennes*, consiste à évaluer rigoureusement ces intégrales pour compter les changements de signe. Malheureusement, le code rigoureux utilisé pour ces évaluations était erroné, et le vrai nombre de cycles limites était en fait 18, au lieu des 26 annoncés – ce qui du coup est inférieur au résultat connu antérieurement $\mathcal{H}(4) \geqslant 22$ **[56]**.

Notre motivation a été d'appliquer les méthodes rigoureuses développées dans cette thèse à l'exemple de T. Johnson afin d'obtenir le plus de zéros possible, toujours rigoureusement !

3 (a) *Trouvons un exemple concret de champ de vecteurs quartique et isolons les cycles limites pour obtenir une nouvelle borne inférieure pour $\mathcal{H}(4)$.*

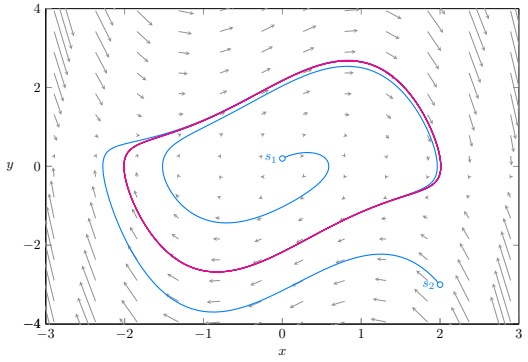
Pour cela, nous réutilisons l'exemple de T. Johnson, représenté sur la **Figure I.7b** :

$$\begin{cases} \dot{x} = -4y^2(y^2 - Y_0) + \varepsilon g_1(x, y), \\ \dot{y} = \quad 4xy(x^2 - X_0) + \varepsilon g_2(x, y), \end{cases}$$

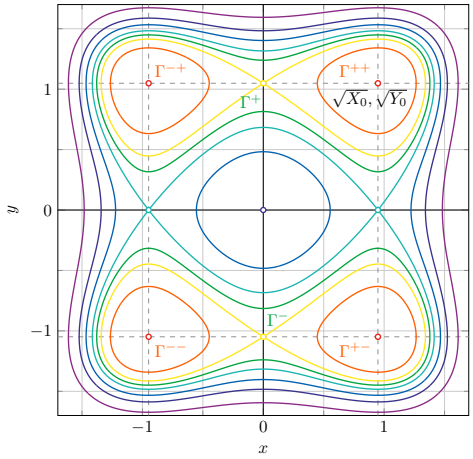
avec $\varepsilon > 0$, et optimisons les coefficients de la perturbation polynomiale $(f, g) \in \mathbb{R}_4[x, y]^2$ afin d'obtenir 24 zéros (ce qui améliore le précédent record $\mathcal{H}(4) \geqslant 22$). Ici, évaluer les intégrales abéliennes revient à intégrer des fonctions algébriques sur des segments compacts, ce qui et possible avec les RPA implémentés dans ChebValid. La **Figure I.7c** montre l'évaluation rigoureuse de l'intégrale abélienne, dont les changements de signe prouve l'existence des cycles limites attendus.

3 (b) *Cette approche peut-elle constituer une preuve* valide *de $\mathcal{H}(4) \geqslant 24$, au sens mathématique ?*

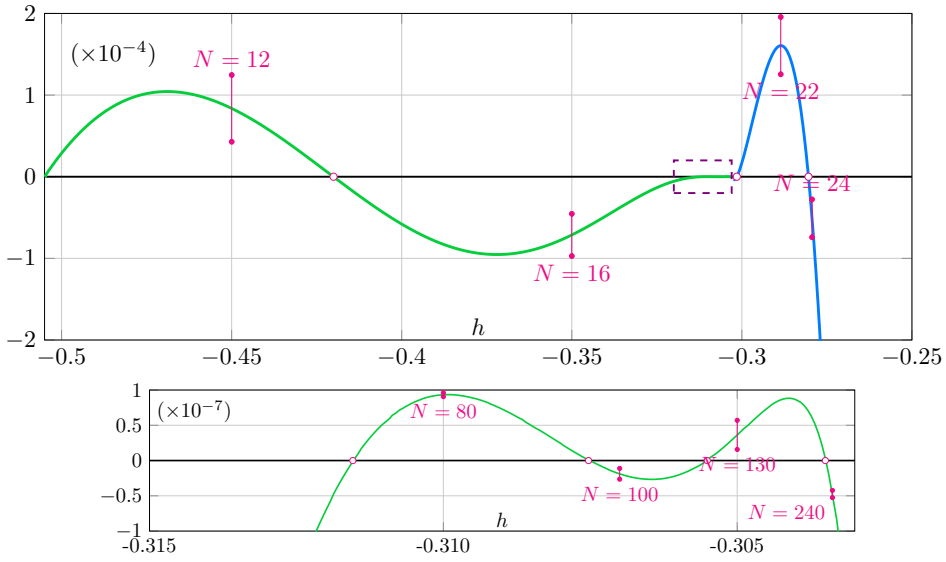
Comme évoqué plus haut, une partie de la communauté mathématique se montre réfractaire aux preuves assistées par ordinateur, et la mésaventure de **[126]** justifie partiellement ces réticences. Heureusement, nous travaillons actuellement au calcul *certifié* de ces intégrales, grâce à notre développement en Coq pour les RPA. Cela permettra donc de donner une preuve définitive de $\mathcal{H}(4) \geqslant 24$.



(a) Exemple de cycle limite avec l'oscillateur de Van der Pol.



(b) Système quartique non perturbé de T. Johnson.



(c) L'évaluation rigoureuse des intégrales abéliennes met en évidence le nombre attendu de changements de signe (en magenta : degré du RPA utilisé et encadrement obtenu pour l'intégrale).

■ **Figure I.7** : Une preuve assistée par ordinateur pour $\mathcal{H}(4) \geqslant 24$ dans le cadre du 16ème problème de Hilbert.

C Sécurité et efficacité pour des applications réelles

L'ingénierie des systèmes critiques renvoie aux applications mettant potentiellement en jeu des vies humaines, ou du moins d'importantes sommes d'argent ou de sérieux risques environnementaux. Sont entre autres concernés les industrie aéronautiques et aérospatiales, les actes médicaux assistées par ordinateur ou les technologies nucléaires. Dans de telles situations, on recourt généralement à d'intensives phases de test afin d'augmenter et de valider le niveau de confiance d'un nouveau logiciel. Néanmoins, en plus de leur coût loin d'être négligeable en temps et en argent, ces tests ne peuvent garantir l'absence de bugs – seulement leur *présence* peut-être détectée. Par conséquent, le besoin de méthodes rigoureuses et formelles a rapidement émergé dans ces domaines, comme en attestent un certain nombre de collaborations entre les mondes industriels et académiques, comme le *NASA Langley Formal Methods Research Program*[28].

Les méthodes développées dans cette thèse ouvrent la voient vers une utilisation systématique des outils rigoureux, symboliques ou formels dans de telles applications de la « vraie vie ». En particulier, je me suis intéressé aux applications aérospatiales. Ce domaine – peut-être en raison des nombreux financements et de l'intérêt public dont il bénéficie, ou d'une législation dans une certaine mesure moins rigide que dans le domaine de l'aéronautique – a déjà été l'objet d'expérimentations intéressantes avec du calcul rigoureux et/ou symbolique-numérique : des modèles de Taylor pour du guidage rigoureux dans **[158]** ; des récurrences D-finies pour l'évaluation rigoureuse de séries de Taylor calculant la probabilité de collision avec des débris spatiaux dans **[227]** ; des séries de Tchebychev multivariées (non rigoureuses) pour la propagation de trajectoires d'un ensemble de débris dans **[255]**, etc.

Le **Chapitre 7**, qui relate une collaboration avec Paulo Ricardo Arantes Gilz et Clément Gazzino, contribue à cette conception de missions spatiales rigoureuses, en validant et utilisant des trajectoires d'engins spatiaux pour des opérations de proximité, c'est-à-dire quand les distances relatives sont faibles en comparaison du rayon de la Terre. L'**Exemple 4** qui suit présente une de ces applications : le problème du *rendez-vous spatial*.

Example 4 : Le problème du rendez-vous spatial

Le problème du rendez-vous spatial consiste, pour un vaisseau actif (par exemple une navette), appelé le *chasseur*, à atteindre une cible passive (par exemple un satellite ou la station spatiale internationale (ISS)), dans un intervalle de temps donné. Pour cela, le chasseur est équipé de propulseurs qui peuvent être activés pour modifier sa trajectoire orbitale actuelle. Cependant, le carburant nécessaire à ces manœuvres n'est disponible qu'en quantité limitée. Lorsque le réservoir est vide, le vaisseau ne peut plus être contrôlé (on le dit *mort*), d'où l'intérêt de minimiser la consommation de carburant pendant les manœuvres orbitales.

Considérons le modèle linéarisé[29] pour la dynamique du mouvement relatif $X = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T \in \mathbb{R}^6$ d'un chasseur par rapport à sa cible :

$$\dot{X}(t) = A(t)X(t) + B(t)u(t), \qquad \textbf{(RDV-i)}$$

où $A(t) \in \mathbb{R}^{6\times6}$, $B(t) \in \mathbb{R}^{p\times6}$, et $u(t) \in \mathbb{R}^p$, qui représente le terme de contrôle pour les poussées, vit dans un espace fonctionnel adéquat. Les poussées sont en général modélisées par

[28] `https://shemesh.larc.nasa.gov/fm/`
[29] Cette hypothèse est généralement admise lors de la phase finale d'un rendez-vous.

des *impulsions*, c'est-à-dire un saut de vitesse instantané plutôt qu'un terme d'accélération. Cela donne le problème du *rendez-vous spatial impulsionnel linéarisé* (voir **Figure I.8a**).

4 (a) *Trouvons les instants t_i et les sauts de vitesse ΔV_i, déterminant la loi de contrôle $u(t)$, pour amener le chasseur d'un état initial donné $X(t_0) = X_0 \in \mathbb{R}^6$ à un état final prescrit $X(t_f) = X_f \in \mathbb{R}^6$ (pour $t_0 < t_f$ fixés), le tout en minimisant la consommation totale de carburant :*

$$\begin{aligned} &\inf_u \|u\|_1 = \inf_u \int_{t_0}^{t_f} \|u(t)\| \mathrm{d}t, \\ t.q. \quad &\dot{X}(t) = A(t)X(t) + B(t)u(t), && t \in [t_0, t_f], \\ &X(t_0) = X_0, \quad X(t_f) = X_f, && t_0,\ t_f \text{ fixés.} \end{aligned} \tag{RDV-ii}$$

Dans les articles **[12, 226]**, auxquels j'ai contribué, nous proposons un algorithme d'échange pour calculer numériquement les instants et valeurs optimaux pour les impulsions. Grosso modo, **(RDV-ii)** y est reformulé comme un problème de *programmation semi-infinie*, c'est-à-dire un problème d'optimisation impliquant un nombre fini de variables réelles mais un nombre infini de contraintes. Partant d'un point infaisable, l'algorithme sélectionne à chaque itération la contrainte la plus violée de sorte à réduire le caractère infaisable du point courant (voir **Figure I.8b**).

Une fois que la loi de contrôle a été calculée numériquement, le cadre rigoureux de la mission spatiale impose de valider la trajectoire résultante du vaisseau, en particulier s'assurer de son état final.

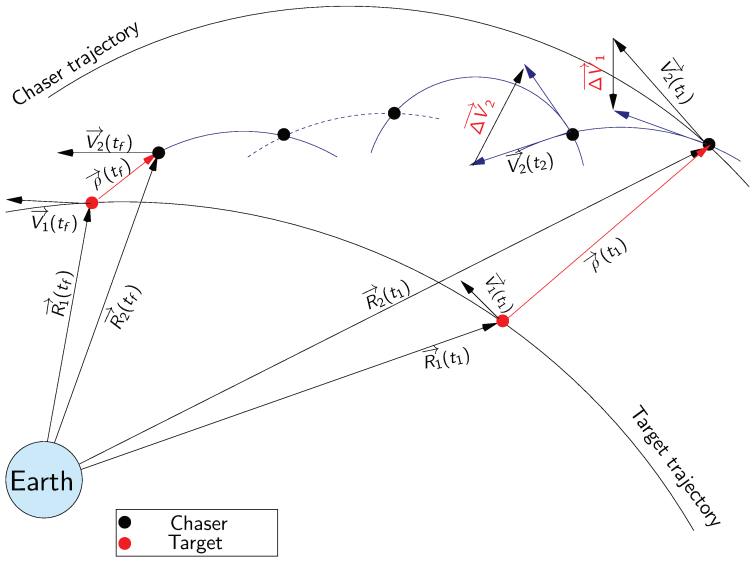
4 (b) *Étant donnés l'état initial X_0, les instants t_i des impulsions et les sauts de vitesse ΔV_i, calculons des encadrements rigoureux for l'état final effectif $\widetilde{X_f}$, et assurons-nous qu'il se trouve suffisamment près de l'état final prescrit X_f.*

En utilisant la méthode de validation du **Chapitre 5** pour les systèmes couplés d'EDOL, nous pouvons calculer des matrices de transition validées sous forme de modèles de Tchebychev pour la dynamique **(RDV-i)**, permettant ainsi de propager rigoureusement l'état $X(t)$ entre deux poussées consécutives. En particulier, cela donne l'encadrement désiré pour l'état final. La **Table I.2** donne les paramètres d'une mission ATV concrète[30] ainsi que les encadrements obtenus pour l'état final dans le plan orbital. Évidemment, cela ne tient pas compte des termes non-linéaires de la dynamique képlérienne, ni des autres perturbations. Néanmoins, la plupart des méthodes de contrôle pour le problème du rendez-vous spatial ne considèrent que des dynamiques linéarisées, de sorte qu'être capable de certifier les résultats calculés numériquement dans ce modèle est déjà un objectif intéressant en soi.

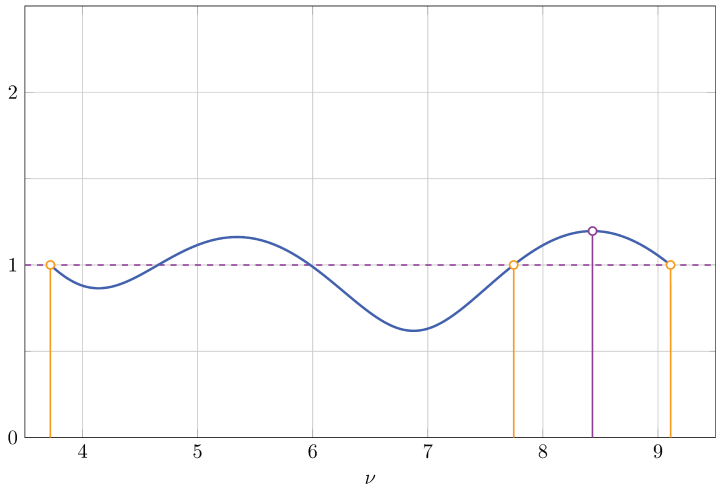
4 (c) *Devons-nous également valider l'optimalité de la loi de contrôle calculée lors de la **Question 4 (a)** ?*

Ce point particulier n'est sans doute pas aussi critique que l'état final – on imagine bien les conséquences d'une navette ATV entrant en collision avec l'ISS. Cependant, si l'on veut être rigoureux jusqu'au bout, il suffit de savoir que l'algorithme présenté dans **[12]** donne à chaque itération un encadrement de la consommation optimale.

[30] *Automated Transfer Vehicle* est le nom du cargo spatial européen utilisé pour le ravitaillement de l'ISS.



(a) Le problème du rendez-vous spatial impulsionnel linéarisé.



(b) L'algorithme d'échange pour le rendre-vous linéarisé.

■ FIGURE I.8 : Résolution numérique du problème de rendez-vous spatial.

Demi-axe majeur :	6763 km
Excentricité :	0.0052
Instant initial :	$\nu_0 = 0$ rad
Instant final :	$\nu_f = 8.1832$ rad
État initial :	$(x, z, \dot{x}, \dot{z}) =$ $(-30, 0.5, 8.514, 0)$ [km – m/s]
État final :	$(x, z, \dot{x}, \dot{z}) =$ $(-100, 0, 0, 0)$ [m – m/s]

(a) Paramètres de la mission.

ν	$\Delta\dot{x}$ [m/s]	$\Delta\dot{z}$ [m/s]
0.0	-7.50230589	0.742372034
1.388128	-1.55579123	0.08834686
6.666595	0.62565013	0.03325936
8.183058	1.06509710	0.11440204

(b) Instants et valeurs des impulsions calculés.

degré	$x(\nu_f)$	$z(\nu_f)$	$\dot{x}(\nu_f)$	$\dot{z}(\nu_f)$
25	-100 + [-2.6861e1, 2.6861e1]	[-7.4084e0, 7.4084e0]	[-1.8133e-2, 1.8133e-2]	[-5.3675e-3, 5.3675e-3]
30	-100 + [-1.0035e-1, 1.0035e-1]	[-2.7676e-2, 2.7676e-2]	[-6.7741e-5, 6.7741e-5]	[-2.0051e-5, 2.0051e-5]
40	-100 + [-2.3194e-5, 2.3190e-5]	[-6.3956e-6, 6.3956e-6]	[-1.5655e-8, 1.5655e-8]	[-4.6336e-9, 4.6336e-9]
50	-100 + [-2.0321e-8, 1.6320e-8]	[-5.0437e-9, 5.0607e-9]	[-1.2358e-11, 1.2376e-11]	[-3.6651e-12, 3.6555e-12]

(c) État final validé, en fonction du degré des polynômes d'approximation.

Table I.2 : Validation *a posteriori* pour une mission ATV dans le plan orbital (x, z).

D Intégrer les aspects « bas niveau »

Le rappel sur la théorie de l'approximation donné dans le **Chapitre 2** suit essentiellement un point de vue purement mathématique : les polynômes sont supposés donnés par des coefficients réels exacts, c'est-à-dire en précision infinie, et l'évaluation $x \mapsto p(x)$ est considérée comme exacte. Certes, les **Chapitres 3**, **4** et **5** bornent rigoureusement les erreurs d'arrondi grâce à l'arithmétique d'intervalles, tout en expliquant dans une certaines mesure comment éviter les phénomènes de surapproximation qui en découlent. Cependant, nous n'y donnons aucune analyse *quantifiée* des erreurs d'arrondis.

Toutefois, certains domaines particuliers exigent une analyse précise des erreurs d'arrondi, comme l'implémentation de fonction en flottants, où la précision du résultat doit être garantie (par exemple pour l'arrondi correct de fonctions **[177]**, voir **Section 1.2**). Dans cette optique, le **Chapitre 8** présente une collaboration avec Mioara Joldes et Denis Arzelier, dans laquelle nous proposons un algorithme d'échange pour calculer des polynômes d'approximation optimisés en tenant compte de l'erreur d'évaluation. Comme le nom le suggère, cet algorithme est très relié avec l'algorithme d'échange de Remez qui calcule des approximations polynomiales uniformes optimales (cf. **Section 2.2.2**), ainsi que l'algorithme d'échange pour les problème du rendez-vous spatial **[12]** mentionné plus haut. Reprenons l'exemple de la fonction d'Airy pour illustrer notre approche.

Example 1 : Fonction d'Airy Ai **– Implémentation en flottants**

Étant donné que Ai ne peut s'exprimer par des fonctions élémentaires, utiliser des polynômes d'approximation est un des outils les plus naturels pour générer des implémentations en flottants. Cela consiste, pour un intervalle $I = [a, b]$ et un degré n fixés, à trouver les coefficients d'un polynôme p dans une base donnée, de telle sorte que $\mathrm{Ai}(x)$ est approximativement calculé comme $p(x)$, pour $x \in I$. Dans un monde idéal sans erreur d'arrondi, ceci s'appelle le problème

minimax, rappelé dans la **Section 2.2.2**

1 (c) *Étant donnés un intervalle compact $I = [a, b]$ et un degré n, trouvons la meilleure approximation polynomiale $p \in \mathbb{R}_n[x]$ pour* Ai*, c'est-à-dire le polynôme p de degré au plus n minimisant l'erreur d'approximation uniforme :*

$$\arg\min_{p\in\mathbb{R}_n[x]} \|\operatorname{Ai} - p\|_{\infty,I} \quad := \quad \arg\min_{p\in\mathbb{R}_n[x]} \max_{x\in I} |\operatorname{Ai}(x) - p(x)|.$$

Ce problème peut être résolu par l'algorithme de Remez **[209, 208, 202]**. En particulier, le caractère équioscillant de l'erreur d'approximation, représenté sur la figure **Figure I.9b**, constitue un preuve d'optimalité.

Cependant, en arithmétique flottante, le polynôme p, en tant que fonction mathématique $x \mapsto p(x)$, est implémentée par une suite d'opérations flottantes, c'est-à-dire un programme $\widetilde{p}$, appelé *schéma d'évaluation*, qui calcule un résultat flottant $\widetilde{p}(x)$ sur une entrée flottante $x \in I$. Dans ce cadre, les coefficients de p sont des flottants, et le résultat $\widetilde{p}(x)$ est sujet à des erreurs d'arrondi. Si p^* désigné le polynôme optimal de la **Question 1 (c)**, et $\widetilde{p^*}$ un schéma d'évaluation où les coefficients de p sont arrondis aux flottants les plus proches, alors il n'y a aucune raison pour laquelle $\widetilde{p^*}$ serait la solution optimale de la question suivante (cf. **Figure I.9c**).

1 (d) *Étant donnés un intervalle compact $I = [a, b]$, un degré n, une précision flottante u et un schéma d'évaluation fixé $p \mapsto \widetilde{p}$ pour des polynômes de degré n, trouvons un polynôme $p \in \mathbb{R}_n[x]$ à coefficients flottants qui minimise l'erreur totale :*

$$\arg\min_{p\in\mathbb{R}_n[x]} \max_{x \text{ FP in } I} |\operatorname{Ai}(x) - \widetilde{p}(x)|.$$

Ceci est une question majeure dans le domaine de l'arithmétique des ordinateurs **[42, 43]**. Notre algorithme constitue une avancée sur la question en optimisant *simultanément* les erreurs d'approximation et d'évaluation. C'est en fait une généralisation de l'algorithme de Remez dans le cadre de la programmation semi-infinie. L'erreur totale de la solution optimale pour cet exemple avec la fonction d'Airy est représentée sur la **Figure I.9d**, où nous pouvons voir que l'erreur d'approximation est « répartie » sur I de telle sorte que les pics les plus élevés se trouvent là où l'erreur d'évaluation est petite.

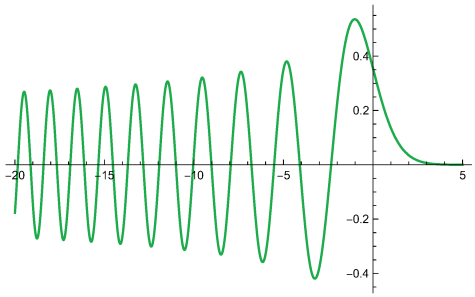
Pour finir, précisions que ce cadre « bas niveau » n'est en rien incompatible avec le calcul rigoureux :

1 (e) *Étant donnée une approximation polynomiale $p \in \mathbb{R}[x]$ pour* Ai*, calculons une borne supérieure sur $I = [a, b]$ pour l'erreur d'approximation ou l'erreur totale.*

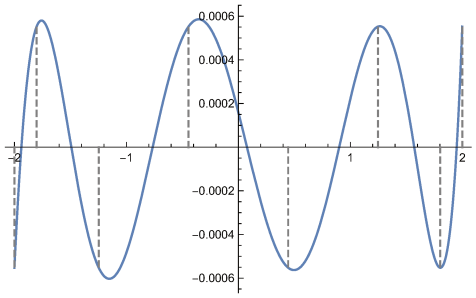
Pour l'erreur d'approximation, cela signifie simplement borner $\|\operatorname{Ai} - p\|_{\infty,I}$. Comme Ai satisfait l'EDOL **(Ai-ii)**, une borne d'erreur rigoureuse peut être calculée en utilisant la méthode de validation du **Chapitre 4**.

En ce qui concerne l'erreur totale, on doit en plus borner l'erreur d'évaluation. Pour cela, on peut utiliser de logiciels certifiés existants, comme Gappa **[67]** qui s'appuie sur la bibliothèque Flocq de Coq mentionnée plus haut.

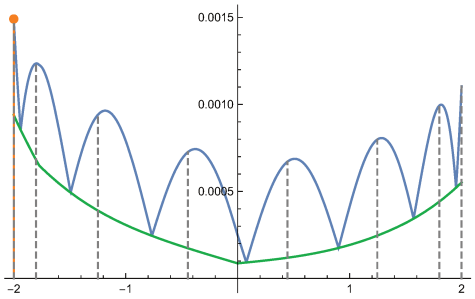
E Nouveaux champs d'application des méthodes rigoureuses et symboliques-numériques



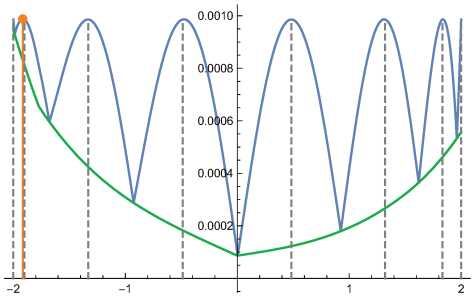
(a) La fonction d'Airy Ai.



(b) Erreur d'approximation pour le polynôme minimax p calculé par l'algorithme de Remez.



(c) Erreur totale pour le polynôme minimax, calculé par l'algorithme de Remez (erreur d'approximation en bleu et erreur d'évaluation en vert).



(d) Erreur totale pour la meilleure approximation p calculée avec notre méthode (erreur d'approximation en bleu et erreur d'évaluation en vert).

■ **Figure I.9** : Problème d'approximation polynomiale pour la fonction d'Airy Ai, avec $I = [-2, 2]$, $n = 6$ et une précision flottante $u = 12$.

Certaines branches des mathématiques ont très tôt su tirer parti du calcul rigoureux, comme les solveurs validés d'équations différentielles ou les optimiseurs rigoureux/certifiés. À l'inverse, certaines applications des méthodes rigoureuses et/ou symboliques-numériques proposées dans ce manuscrit mettent en lumière de nouvelles interactions entre les mathématiques traditionnelles et le calcul rigoureux.

- Dans le**Chapitre 9**, nous (c'est-à-dire Mioara Joldes, Jean-Bernard Lasserre et moi-même) proposons de nouveaux algorithmes pour résoudre des problèmes inverses sur les mesures, autrement dit pour reconstruire le support et la densité d'une mesure à partir d'un nombre fini de ses moments. Pour ce faire, nous tirons profit de la notion de D-finitude (et de celle, étroitement relié, d'holonomie) mentionnée plus haut. Bien que non rigoureux, ces méthodes peuvent être qualifiées de *symboliques-numériques* : le cadre algébrique des équations D-finies produit des systèmes linéaires résolus *in fine* numériquement, puisque les moments donnés ne sont en général connus qu'en précision finie. Illustrons la méthode avec l'exemple suivant.

Example 5 : Moments d'une densité gaussienne sur un ensemble semi-algébrique

Soit $f(x,y) := \exp(a_{xx}x^2 + a_{yy}y^2 + a_{xy}xy + a_x x + a_y y + a_1)$ une densité gaussienne (non normalisée) dans le plan et G l'ensemble semi-algébrique de dimension pleine dans $\mathbb{R}^2$ représenté sur la **Figure I.10** par la région hachurée, dont le bord est contenu dans l'ensemble des zéros du polynôme :

$$g(x,y) := (x^2+y^2-9)(x^2+y^2-1)((x-2)^2+y^2-1)(x^2+(y-2)^2-1).$$

Les moments $m_{\alpha,\beta}$ de la mesure $\mu := f\mathbb{1}_G$ (mesure gaussienne restreinte à G) sont définis de la manière suivante :

$$m_{\alpha,\beta} := \int_G x^\alpha y^\beta f(x,y)\mathrm{d}x\mathrm{d}y.$$

Le **Chapitre 9** répond aux deux questions suivantes. Considérons d'abord le *problème direct*.

5 (a) *Étant donnés la densité f et le polynôme g pour le bord, déduire un ensemble « complet »*[31] *de récurrences pour les moments $m_{\alpha,\beta}$, permettant de calculer tous les moments à partir d'un nombre fini d'entre eux.*

Cette question est en partie résolue dans la **Section 9.3** en se basant sur une heuristique, avec des résultats plus précis dans le cas d'une densité de type exponentielle de polynôme, incluant les densités gaussiennes. Cela peut-être vu comme une alternative aux techniques usuelles de *Télescopage Créatif* (voir par exemple **[188]**) dans le cas d'un support semi-algébrique.

À l'inverse, voici le *problème inverse*.

5 (b) *Étant donné un nombre fini de moments $m_{\alpha,\beta}$, essayons de retrouver le polynôme dans l'exponentielle définissant f, ainsi qu'un polynôme non trivial dont l'ensemble des zéros contient le bord de G.*

Des algorithmes pour résoudre ce problème sont donnés dans la **Section 9.4**. Ils consistent à « deviner » des récurrences pour les moments en résolvant des systèmes linéaires impliquant

[31] Le terme technique adéquat serait ici *holonomique* (voir la **Section 2.1.2**).

suffisamment de moments connus, permettant ainsi de reconstruire le polynôme pour le bord et des équations différentielles satisfaites par la densité D-finie f (voir la **Figure I.10**). Dans cet exemple exponentielle de polynôme, le nombre de moments nécessaires peut être borné *a priori*.

- Les modèles de Tchebychev, largement promus dans cette thèse, constituent un outil puissant pour les fonctions analytiques sur des segments compacts. Cependant, de telles hypothèses ne sont pas tout le temps satisfaites : l'intervalle considéré peut être ouvert en raison de singularités, ou même non borné pour des problèmes à « horizon infini ». Par conséquent, c'est un défi pertinent que d'étendre les approximations rigoureuses à des fonctions non polynomiales (par exemple, Hermite ou Bessel), ou même des familles non linéaires (par exemple, les fractions rationnelles). C'est à l'heure actuelle un problème insuffisamment traité dans la littérature et constitue une de mes orientations de recherche dans un futur proche. La question suivante traités dans le **Chapitre 6** illustre parfaitement ce besoin de tels RPA généralisés.

Example 3 : Calculer des cycles limite dans le 16ème problème de Hilbert

3 (c) *Est-il possible de prouver rigoureusement que l'exemple particulier de T. Johnson ne peut donner plus de 24 cycles limite ?*

Pour prouver qu'aucune combinaison linéaire des intégrales abéliennes considérées ne peut produire plus de zéros, nous recourons à la notion de *système de Tchebychev* (définie dans le **Chapitre 2**), dans la **Section 6.4**. Plus précisément, il nous faut calculer le wronskien d'un système d'intégrales abéliennes. Malheureusement, ce wronskien possède l'asymptotique suivante en 0, pour $t > 0$:

$$W(t) = \sum_{n \geqslant n_1} a_n t^n + \log(t) \sum_{n \geqslant n_2} b_n t^n.$$

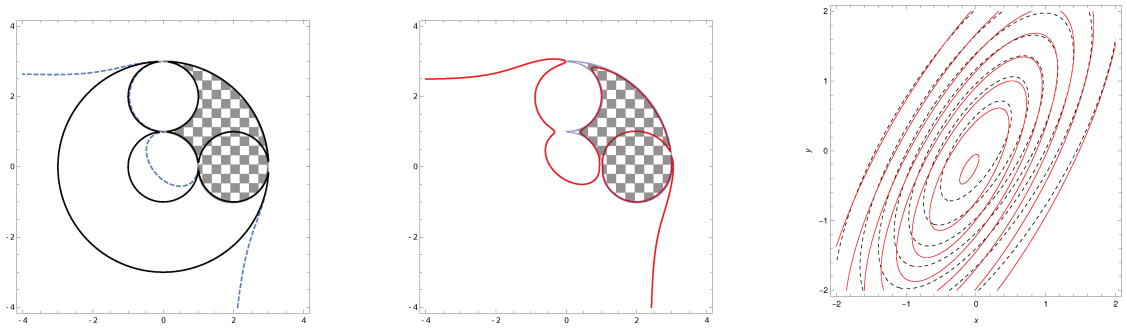
avec $n_1, n_2 \in \mathbb{Z}$. Les techniques D-finies et la transformée de Laplace symbolique vont par chance nous aider à calculer (un nombre fini de) ces coefficients a_n, b_n. Nous devons également développer de nouveaux outils rigoureux pour borner le reste – les difficultés provenant des termes en $\log(t)$ et des puissances négatives de t. Ainsi, cet exemple illustre une situation où des RPA *généralisés* seraient utiles.

F Concevoir des implémentations *open source*

En plus des contributions théoriques mentionnées tout au long de cette introduction, mon travail a également consisté à développer plusieurs implémentations. Bien qu'encore expérimentales, ces implémentations jouent un rôle important dans l'approche rigoureuse et algorithmique des problèmes fonctionnels défendue dans cette thèse, à base de RPA et d'outils symboliques-numériques.

- J'ai développé une bibliothèque C *open source*, que l'on appellera ChebValid[32], qui implémente l'arithmétique des modèles de Tchebychev détaillée dans le **Chapitre 3**, ainsi

[32] disponible sur `https://gforge.inria.fr/projects/tchebyapprox/`



(a) Reconstruction (en pointillés bleus) du bord algébrique (en noir) du support (hachuré).

(b) Reconstruction (en rouge) du support, à partir de moments donnés moins précisément.

(c) Reconstruction (en pointillés bleus) d'une densité gaussienne (en orange).

■ **Figure I.10** : Exemple de reconstruction de la densité et du support à partir des moments, extrait du **Chapitre 9**.

que les algorithmes de validation pour les solutions d'EDOL des **Chapitres 4** et **5**. Au delà des exemples illustratifs donnés dans ces chapitres, cette bibliothèque a également été utilisée avec succès dans les **Chapitres 6** et **7** pour le dénombrement rigoureux de cycles limites (**Exemple 3**) et la validation de trajectoires d'engins spatiaux (**Exemple 4**).

- Avec Assia Mahboubi et Damien Pous, nous avons développé une formalisation en Coq[33] **[40]** des RPA avec opérations arithmétiques, munie d'une implémentation complète sur les modèles de Tchebychev. Le **Chapitre 3** en donnera plus de détails. Les intégrales abéliennes de l'**Exemple 3** (cf. **Chapitre 6**), dans un premier temps calculées avec ChebValid, ont dans un second temps été recalculées avec les modèles de Tchebychev de ce développement Coq, offrant ainsi le plus haut niveau de confiance pour une preuve assistée par ordinateur.

- Mioara Joldes et moi-même ont également prototypé plusieurs scripts en Maple et Mathematica pour les problèmes traités dans les **Chapitres 8** et **9**. Pour le premier, (obtention de polynômes d'approximation avec erreur d'évaluation optimisée, voir **Exemple 1**), un script Mathematica complet est disponible ici[34]. Pour le second (voir **Exemple 5**), des scripts Maple pour les différents exemples donnés dans le **Chapitre 9** sont disponibles ici[35].

[33] disponible sur `http://perso.ens-lyon.fr/florent.brehard/chebapprox/`
[34] disponible sur `http://perso.ens-lyon.fr/fbrehard/EvalMinimax`
[35] disponible sur `http://homepages.laas.fr/fbrehard/HolonomicMomentProblem`

Part I

Rigorous Polynomial Approximations from Scratch: Selected Topics and Personal Contributions

Computing with Numbers

1

Les hommes sont comme les chiffres : ils n'acquièrent de valeur que par leur position.
— Napoléon Bonaparte
On gouverne mieux les hommes par leurs vices que par leurs vertus.
— idem

At the core of scientific computing is the calculation with numbers, where "*numbers*" usually refer to elements of the real line $\mathbb{R}$ or of the complex plane $\mathbb{C}$. A wide range of works in applied mathematics or numerical analysis, targeting problems from, for instance, linear algebra, functional analysis or optimization, assume that a basic arithmetic on those "*numbers*" is available. Yet, this point of view is not fully satisfactory when designing and implementing algorithms, since the way numbers are represented and computed with, may have a major impact on the result accuracy and time complexity.

Although this thesis mainly targets algorithms executed on computers, it is worth noting that the problem of representing numbers has been raised since the earliest days of mathematics. Indeed, at the time of by-hand computations, efficient arithmetic operations were certainly not an unnecessary luxury. Conversely, the power of current computers may result in these computations being taken for granted by some engineers or numerical analysts. Yet, this question becomes central again for intensive computations or – closer to our considerations – rigorous numerics.

After a brief overview of *exact* representation of numbers, we present the *floating-point* numbers, which are more or less ubiquitous in scientific computing, due to their ability to efficiently approximate real numbers and operations. This finally leads us to *interval arithmetic*, which is to rigorous numerics what floating-point numbers are to numerical analysis: an essential low-level building block.

1.1 About exact representations of numbers

It seems rather natural to start with an *exact* representation of numbers, which means that numbers are in a 1-1 correspondence with finite sequences of symbols (or bits, when working on computers). Obviously, the whole real line $\mathbb{R}$ cannot be represented that way, for it is an *uncountable* set. However, a wide range of computations actually take place in specific *countable* subsets for which exact representations are available. We here provide rather standard examples.

1.1.1 ▶ Natural numbers

— Who wouldn't start with them?

Integers are usually employed in a very intuitive and informal manner, even by most mathematicians. This is perfectly sufficient in most cases. For our purpose, it is however relevant to focus on some aspects of their definition and representation. The few historical remarks below reflect my own limited knowledge in this fascinating epistemological topic. An extensive survey is given in the wide-public book **[115]**, supplemented by a second book **[114]** focusing more specifically on automated computation.

Set theory

Let us first recall the construction of natural numbers as *von Neumann ordinals* inside *set theory*, which constitutes the most classical foundations of mathematics **[256]**, **[109,** Chap. 3**]**.

Definition 1.1 (Von Neumann natural numbers) *In set theory, natural numbers are inductively constructed from the empty set using set theoretic operations:*

$$0 := \varnothing, \qquad \text{and} \qquad n+1 := n \cup \{n\} = \{0, 1, \dots, n\}, \quad n \geqslant 0.$$

The standard comparison relation is defined by membership:

$$n < m \quad \Leftrightarrow \quad n \in m.$$

Table 1.1 lists the encodings of the first natural numbers following **Definition 1.1**. Clearly, the set-theoretic encoding of n has exponential size with respect to n and will never be used in practice on computers.

Unary system

Another possible encoding is the *unary numeral system*, in which a natural number n is represented by a sequence of n copies of a unique symbol. Bigger quantities, like powers of 10, can be grouped using new symbols. This quite natural encoding is how children usually learn

$$
\begin{array}{rcl}
0 & = & \varnothing \\
1 & = & \{\varnothing\} \\
2 & = & \{\varnothing,\{\varnothing\}\} \\
3 & = & \{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\}\} \\
4 & = & \{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\},\{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\}\}\} \\
5 & = & \{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\},\{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\}\}, \\
 & & \{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\},\{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\}\}\}\} \\
6 & = & \{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\},\{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\}\}, \\
 & & \{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\},\{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\}\}\}, \\
 & & \{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\},\{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\}\}, \\
 & & \{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\},\{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\}\}\}\}\} \\
7 & = & \{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\},\{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\}\}, \\
 & & \{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\},\{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\}\}\}, \\
 & & \{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\},\{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\}\}, \\
 & & \{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\},\{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\}\}\}\}, \\
 & & \{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\},\{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\}\}, \\
 & & \{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\},\{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\}\}\}, \\
 & & \{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\},\{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\}\}, \\
 & & \{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\},\{\varnothing,\{\varnothing\},\{\varnothing,\{\varnothing\}\}\}\}\}\}\}
\end{array}
$$

Table 1.1: Set-theoretic encodings of the first eight natural numbers

to count. It is moreover a quite practical system to count things incrementally. The *tally sticks* **[263]** **[115,** Chap. 5**]**, which were animal bones on which notches were carved to count objects, money or events during the Paleolithic, are one of the first examples of the use of the unary numeral system. Also, the Moscow Mathematical Papyrus is an ancient Egyptian papyrus adressing common problems in mathematics and uses the unary system **[115,** Chap. 14**]** (a short illustrated introduction to Egyptian numerals may be found at the following webpage[1] **[31]**).

This linear encoding reflects the axioms of Peano's arithmetic **[253,** Peano (1989)**]**: there is one 0 (the empty sequence) and the successor operation, obtained by adding one copy of the symbol to the sequence, is injective and 0 does not belong to its image. The standard implementation of natural numbers in the Coq formal proof language closely follows this approach. The inductive type `nat` is built from two constructors: a constant `O : nat` for 0 and a unary `S : nat → nat` for the successor. The benefit of such a representation is that the induction principle automatically associated to this inductive type matches the usual *induction principle* on natural numbers **[162]**:

```
Inductive nat : Set :=
    | O : nat
    | S : nat → nat.

nat_rect : ∀ P : nat → Type,
    P 0 → (∀ n : nat, P n → P (S n))
    → ∀ n : nat, P n
```

[1] `http://arindambose.com/?p=737` "Did You Know : The History of Egyptian Mathematics (Part II) – Egyptian Numerals." by Arindam Bose.

Positional systems

Throughout history, *positional numeration systems* [115, Chap. 2] have been adopted by more and more civilizations under different forms, the *decimal system* being used almost everywhere on Earth nowadays [2]. The main idea consists in choosing a radix $b \geqslant 2$ and a set Σ_b of b distinct symbols that is identified with $\{0, 1, \ldots, b-1\}$. Then, a sequence $a_n a_{n-1} \ldots a_1 a_0$ of symbols in Σ_b corresponds to the number $N = \sum_{k=0}^{n} a_k b^k$. Each nonzero natural number N admits a unique representation with nonzero leftmost symbol, of length $\lfloor \log_b N \rfloor + 1$. This logarithmic-size encoding and corresponding logarithmic-time arithmetic operations (addition, subtraction, multiplication) account for the dominance of positional systems. While the decimal system that we use in everyday life uses radix 10, most computer systems are based on radix 2.

It is important to note that the (signed or unsigned) integers available at the level of the processor have *fixed size* (usually 8 bits for `char`, 16 for `short`, 32 for `int`, 64 for `long`...). In fact, they implement *modular arithmetic* rather than Peano's arithmetic. When working with large numbers for which overflows (the memory capicity is exceeded) may occur, one should use *arbitrary-precision* integer libraries, which implement numbers as finite bit sequences of arbitrary size at software level. One of the most widely used such ones is the C *GNU Multiple Precision Arithmetic Library* [97] (GNU MP or GMP for short, used for example in the computer algebra systems MAPLE[3] and MATHEMATICA[4]), providing the `mpz_t` type for integers. An example code to compute the factorial function using GMP is given in Table 1.2. The result does not fit into a 64-bit machine integer.

1.1.2 ▶ Rational numbers

Built on integers, rational numbers also admit exact representations on computers. They are implemented by the `mpq_t` type in the GMP library [97].

Definition 1.2 (Rational numbers) *The set $\mathbb{Q}$ of rational numbers is the* field of fractions *associated to the ring of integers. Two fractions $\frac{n_1}{d_1}$ and $\frac{n_2}{d_2}$ (with $d_1, d_2 \neq 0$) represent the same rational number if and only if $n_1 d_2 = n_2 d_1$. The canonical fraction for $q \in \mathbb{Q}$ is by convention $q = \frac{n}{d}$ with (n, d) coprime and $d > 0$ (in particular, the canonical representation of 0 is $\frac{0}{1}$).*

Arithmetic operations are defined as follows:

$$\begin{aligned} -\frac{n_1}{d_1} &:= \frac{-n_1}{d_1}, & \frac{n_1}{d_1} + \frac{n_2}{d_2} &:= \frac{n_1 d_2 + n_2 d_1}{d_1 d_2}, \\ \frac{n_1}{d_1} \times \frac{n_2}{d_2} &:= \frac{n_1 n_2}{d_1 d_2}, & \frac{n_1}{d_1} \div \frac{n_2}{d_2} &:= \frac{n_1 d_2}{d_1 n_2} \ (n_2 \neq 0). \end{aligned} \tag{1.1}$$

A wide range of computations in numerical analysis take place inside $\mathbb{Q}$, e.g. solving linear systems, interpolating rational values at rational points, solving linear optimization problems

[2] This decimal dominance concerns not only numbers, but also most physical units. The French Revolution played a decisive role in that domain, despite the significant failure of decimal time division – not to mention the reluctance of the "*perfide Albion*" to adopt the metric system.

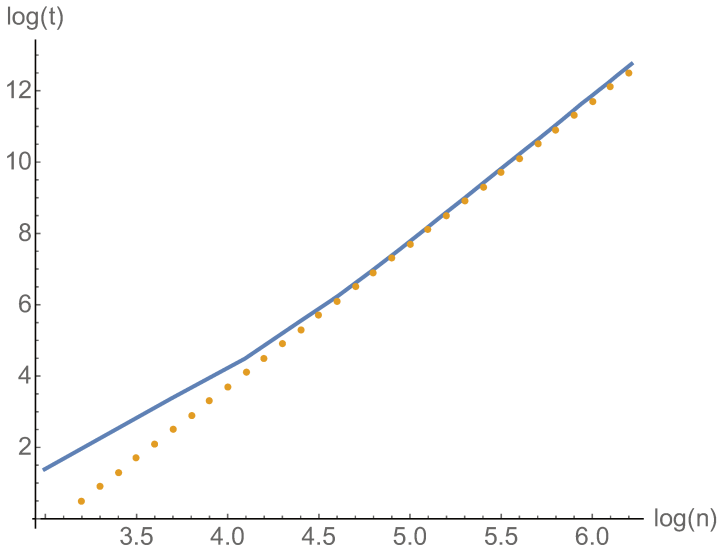
[3] "The GNU Multiple Precision (GMP) Library", *Maplesoft*: `https://fr.maplesoft.com/support/help/AddOns/view.aspx?path=GMP`

[4] "Some Notes on Internal Implementation", *Wolfram Mathematica*: `https://reference.wolfram.com/language/tutorial/SomeNotesOnInternalImplementation.html`

```c
#include<stdio.h>
#include<gmp.h>

void factorial(mpz_t rop, mpz_t op)
{
  // init local variables
  mpz_t fact, count;
  mpz_init_set_ui(fact, 1);
  mpz_init_set_ui(count, 2);

  // compute the factorial
  while (mpz_cmp(count, op) <= 0) {
    mpz_mul(fact, fact, count);
    mpz_add_ui(count, count, 1);
  }

  // assign result and clear variables
  mpz_set(rop, fact);
  mpz_clear(fact);
  mpz_clear(count);
}

int main()
{
  mpz_t n, fact_n;
  mpz_init_set_si(n, 37);
  mpz_init(fact_n);

  // compute 37! and print result
  factorial(fact_n, n);
  mpz_out_str(stdout, 10, fact_n);
  printf("\n");

  mpz_clear(n);
  mpz_clear(fact_n);

  return 0;
}
```

Table 1.2: Example C code using GMP to implement the factorial. The output of this program is: 13763753091226345046315979581580902400000000.

n	t (ms)	n	t (ms)
		260	23485
20	4	280	31793
40	29	300	41902
60	89	320	54674
80	241	340	70367
100	517	360	89652
120	1033	380	112968
140	1892	400	138172
160	3241	420	168552
180	5249	440	202921
200	8009	460	245397
220	11867	480	291881
240	16966	500	345001



■ **Figure 1.1**: Timings (in ms) of a Gaussian inversion procedure applied to H_n, using GMP's rational numbers. The oblique asymptote (in dashed yellow) on the log-log plot suggests a time complexity of $\mathcal{O}(n^\alpha)$ with $\alpha \geqslant 4$.

with rational input, etc. It is therefore tempting to always perform such computations in *exact rational arithmetic*, where rational numbers are represented by a pair of integers (numerator and denominator), and the arithmetic operations are done as in Eq. (1.1). However, the size of the integers rapidly becomes an issue. Indeed, if $r_i = \frac{n_i}{d_i}$ for $i \in \{1, 2\}$ with n_i, d_i represented using d bits, then the size of the numerator and denominator of $r_1 * r_2$ (for $* \in \{+, \times, \div\}$) can reach $2d-1$ bits (a gcd algorithm can be used to reduce the fractions obtained by Equation (1.1), but this does not improve the worst-case analysis). Therefore, beside the arithmetic complexity, where a constant cost is assumed for all arithmetic operations, sometimes, bit complexity is needed to provide more realistic time estimates for algorithms using exact rational numbers.

■ **Example 1.3** *The Hilbert matrix* $H_n := \left(\frac{1}{i+j-1}\right)_{1\leqslant i,j\leqslant n}$, *appearing in least-squares approximation of functions with polynomials, is well-known to be particularly* ill-conditioned *(see* [17] *and references therein). Inverting this matrix using machine floating-point arithmetic may lead to huge numerical errors, as it will be observed in* ***Example 1.10****. Therefore, using exact rational arithmetic seems to be the appropriate solution. However, as depicted on* ***Figure 1.1****, the computation time of a standard Gaussian inversion procedure applied to* H_n *grows faster than the arithmetical complexity, which is* $\mathcal{O}(n^3)$.

1.1.3 ▶ Algebraic numbers

Furthermore, not all computations take place in the field of rational numbers. Since $\mathbb{Q}$ is dense in $\mathbb{R}$, one can *approximate* arbitrary real numbers by rational ones. But when *exact* computations are required, one needs to find good representations of larger classes of numbers for which *algorithms* are available for arithmetic operations. An important such class is the

field $\overline{\mathbb{Q}}$ of *algebraic numbers* over $\mathbb{Q}$ **[105,** Chap. 4**]**.

Definition 1.4 (Algebraic number) *A real (or complex) number a is algebraic over $\mathbb{Q}$ if there exists a nonzero polynomial P with rational coefficients such that $P(a) = 0$.*

Theorem 1.5 ($\overline{\mathbb{Q}}$ is a subfield of $\mathbb{C}$ **[105,** Thm. 50**]**) *If a and b are algebraic, then so are $a + b$, $a - b$, ab and, if $b \neq 0$, a/b.*

The fact that $\overline{\mathbb{Q}}$ is closed under these operations directly follows from the observation that a is algebraic if and only if $\{a^n, n \in \mathbb{N}\}$ spans a finite-dimensional $\mathbb{Q}$-vector space. Moreover, the corresponding annihilating polynomials are efficiently computed using the notion of resultant **[257, 236,** Chap. 6**]**.

These operations on polynomials allow for an algorithmic arithmetic for algebraic numbers. An algebraic number $a \in \overline{\mathbb{Q}}$ is represented by a polynomial $P \in \mathbb{Q}[x]$ such that $P(a) = 0$ together with an interval $\mathbb{a} = [\underline{a}, \overline{a}] \subseteq \mathbb{R}$ ($\underline{a}, \overline{a} \in \mathbb{Q}$) such that a is the only root of P in $\mathbb{a}$. Arithmetic operations are performed by applying the above transforms on polynomials and the corresponding operations on intervals (defined in **Section 1.3**). As explained in **[236]**, the interval enclosures $\mathbb{a}$ of a and $\mathbb{b}$ of b must sometimes be made tighter to ensure that the interval enclosure $\mathbb{a} \circledast \mathbb{b}$ of $a * b$ contains only one root of the resulting polynomial (see **Example 1.29**). Such a representation of algebraic numbers is:

- *exact* because to one pair $(P, \mathbb{i})$ can correspond only one value, i.e., the algebraic number under consideration, and this property is maintained by the arithmetic operations;
- *effective* because arithmetic operations are given by algorithms, and (in)equalities are decidable **[236]**.

Beyond exact representations. In the above paragraphs, *exact* and *effective* representations for certain classes of real and complex numbers have been given. They offer the advantage of performing exact computations (that is, without numerical error). However, as illustrated by **Example 1.3**, the bit size of such representations tends to rapidly grow during the computation, inducing substantial space and time complexities.

The next section focuses on floating-point arithmetic, where exactness is sacrificed to practical efficiency. Interval arithmetics will be presented in **Section 1.3** as a compromise between efficiency and reliability, where *exact* is replaced by *rigorous* or *set-valued* numerics, meaning that a computed interval necessarily *contains* the correct mathematical value.

1.2 Floating-point arithmetic

The *floating-point numbers* are one of the most popular formats to *approximately* represent real numbers, while avoiding the complexity issues discussed in the previous section. Roughly speaking, they consist in keeping only a finite number of bits in the binary expansion of a number. In the early days of computers, each hardware implementation used its own specification of floating-point arithmetic. This led to different results for the same computation performed on different architectures, violating the principle of *reproducibility*.

In response to this increasingly chaotic situation, the IEEE 754-1985 standard for floating-point arithmetic [112] was born in 1985 out of a joint effort between industry actors and academic research. Since then, a new version of the standard (IEEE 754-2008) was adopted in 2008 [113]. The interested reader can find a thorough introduction to floating-point arithmetic as defined by this standard in [177]. This section focuses on the most basic definitions as well as the notion of *rounding errors*, which are a direct consequence of the discretization of the real line induced by floating-point numbers.

1.2.1 ▶ Floating-point numbers

A reference definition of floating-point numbers is given in [177, Sec. 2.1.1].

Definition 1.6 (Floating-Point Numbers) *Given:*

- *an integer radix* $\beta \geqslant 2$ *(*$\beta = 2$ *in most implementations),*
- *an integer precision parameter* $p \geqslant 2$*,*
- *and extremal exponents* $e_{\min}, e_{\max} \in \mathbb{Z} \cup \{-\infty, +\infty\}$ *such that* $-\infty \leqslant e_{\min} < 0 < e_{\max} \leqslant +\infty$*,*

a triple (s, m, e) *represents the floating-point number* x *if:*

$$x = (-1)^s \cdot m \cdot \beta^{e-p+1},$$

where:

- $s \in \{0, 1\}$ *is the* sign bit*,*
- e *is the (integer)* exponent *satisfying* $e_{\min} \leqslant e \leqslant e_{\max}$*,*
- m *is the* integral significand *belonging to the set* $M = [\![0, \beta^p - 1]\!]$ *of nonnegative integers representable in radix* β *using* p *"digits". The* normal significand $m \cdot 2^{-p+1}$ *has one "digit" before the point and* $p - 1$ *after:*

$$m \cdot 2^{-p+1} = m_0.m_1 m_2 \dots m_{p-1} \leqslant \beta.$$

We denote by $\mathbb{F}_{\beta,p}^{e_{\min},e_{\max}}$ *the set of floating-point numbers representable in this way:*

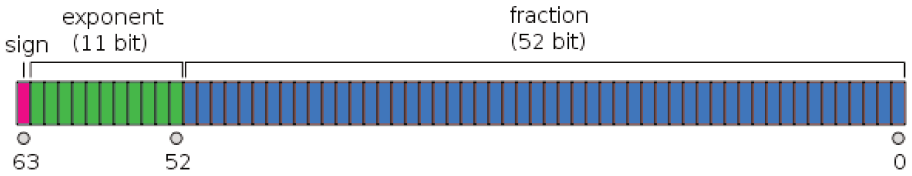
$$\mathbb{F}_{\beta,p}^{e_{\min},e_{\max}} = \{x = (-1)^s \cdot m \cdot \beta^{e-p+1}, s \in \{0, 1\}, e \in [\![e_{\min}, e_{\max}]\!], m \in M\}.$$

From **Definition 1.6**, it appears that two different triples may represent the same floating-point number. By convention, the first "digit" of the integral significand m must be nonzero, and *normalizing* a floating-point number in this way is always possible by reducing the exponent e, unless we reach the minimal exponent $e_{\min}$. This results into the following two classes of floating-point numbers [177, Sec. 2.1.2]:

- *normal numbers* with $e \in [\![e_{\min}, e_{\max}]\!]$ and $m \in [\![\beta^{p-1}, \beta^p - 1]\!]$;
- *subnormal numbers* with $e = e_{\min}$ and $m \in [\![0, \beta^{p-1} - 1]\!]$.

The presence of subnormal numbers allows for a uniform filling of the gap between 0 and the smallest normal floating-point number $2^{e_{\min}}$.

For the sake of simplicity, we omit to explicitly mention the exponent range $[\![e_{\min}, e_{\max}]\!]$ as well as the precision p. We therefore simply denote by $\mathbb{F}$ a given binary floating-point system.



(a) Physical representation of a floating-point number, exemplified with the IEEE 754 binary64 format.

Name *(precision)*	binary16 —	binary32 *(single)*	binary64 *(double)*	binary128 *(quad)*
p	11	24	53	113
$e_{\min}$	-14	-126	-1022	-16382
$e_{\max}$	$+15$	$+127$	$+1023$	$+16383$

(b) Exponent range $[e_{\min}, e_{\max}]$ and precision p in the main formats specified by the IEEE 754-2008 standard.

Figure 1.2: IEEE 754-2008 floating-point format

Standard floating-point formats and implementations

Besides providing general definitions for floating-point arithmetics, the last release of the IEEE 754 standard (IEEE 754-2008 [113]) also fixes standard formats dedicated to hardware implementation. A floating-point number is represented as a sequence of bits encoding the sign, the exponent and the significand, which usually matches the size of a register in a processor, or a multiple of it (see **Figure 1.2a**). These different formats are listed in **Table 1.2b** and reflect the evolution of the registers' size in recent processors.

Simultaneously, software libraries were developed to build upon the features of hardware floating-point arithmetic. Although extremely efficient, hardware implementations are usually (except for ad-hoc custom-built processors) limited to basic arithmetic operations (addition, multiplication, division, square root) in several specified formats, so custom precisions or more complicated functions need to be implemented in software. Let us mention some of these software libraries, and the extended features they offer.

- Although correct rounding of elementary functions is recommended by the IEEE 754-2008 standard, most hardware implementations only guarantee that $+, -, \times, /, \sqrt{}$ and the *fused multiply-add* (FMA) $a, b, c \mapsto a + b \times c$, are correctly rounded. To overcome these limitations, *mathematical libraries* (often referred to as *libms*) were developed. In particular, the CRLIBM library [64] provides correct rounding for most elementary functions: exp, ln, $\log_2$, sin, cos, tan, arctan, ... A strongly related problem is the famous *Table Maker's Dilemma* (TMD) [177, Chap. 10], which can be roughly summarized in the following question: For a given floating-point precision p and an elementary function f over a fixed interval I, to which accuracy do we need to compute $f(x)$ for $x \in I$ a floating-point number, in order to return the correctly rounded result $\widetilde{f(x)}$ in precision p, in the worst case?

- The double-precision format available on most current architectures may seem sufficient for most applications, since, as mentioned in [177], "one should never forget that with

50 bits, one can express the distance from the Earth to the Moon with an error less than the thickness of a bacterium". Nevertheless, some particularly *ill-conditioned* problems (see **Example 1.10**) require a higher floating-point precision. In such cases and despite their lower efficiency, resorting to *multiple-precision* floating-point libraries may be necessary. One of the most popular is MPFR **[83]**, which provides highly optimized correctly rounded elementary functions to arbitrary user-defined precision.

- A more recent trend consists in taking advantage of the parallel architectures, such as Graphics Processing Units (GPUs), for *high performance computing* (HPC). Several libraries were recently developed to port multiple-precision floating-point arithmetic to GPUs. The particularly promising CAMPARY library[5] **[130, 201]** implements arithmetic operations using *floating-point expansions*, that is, unevaluated sums of floating-point numbers, within arbitrary precision and with proved error bounds. This is a major step toward efficient multi-precision computing, and, closer to our interest, efficient interval arithmetic (**Section 1.3**).

1.2.2 ▶ Rounding modes and rounding errors

Replacing real numbers by floating-point numbers necessarily induces a discretization of the real line (see **Figure 1.3**). Whereas each floating-point number can be seen as a real number in a unique way, the converse is not true: most of the real numbers cannot be represented as elements of $\mathbb{F}$. Hence, numerical errors are an intrinsic limitation of floating-point computation, even for the basic arithmetic operations such as addition or multiplication.

A relevant notion to estimate these numerical errors is the *unit in the last place* (ulp), defined as follows **[177,** Def. 2.6**]**:

Definition 1.7 (Unit in the last place – Goldberg's definition) *For a nonzero real number x, let $e \in \mathbb{Z}$ such that $2^e \leqslant |x| < 2^{e+1}$. Then:*

$$\mathrm{ulp}(x) = 2^{\max(e, e_{\min}) - p + 1}.$$

If x is a (normal or subnormal) floating-point number, then $\mathrm{ulp}(x)$ is the power of two represented by the right-most digit of its significand.

The unit in the last place corresponds to the distance between two consecutive floating-point numbers in one binade, that is with the same exponent (See **Figure 1.3**). It is particularly useful to bound the effect of rounding numbers, as it will appear below.

Rounding modes and errors

The operation of replacing a real number by an *approximating* floating-point number is called *rounding* **[177,** Sec. 2.2**]**. The IEEE 754 standard defines four rounding modes:

- Round towards $-\infty$: $\mathbf{RD}(x)$ is the largest floating-point number smaller or equal to x, possibly equal to the exceptional value $-\infty$:

$$\mathbf{RD}(x) = \max\{y \in \mathbb{F}, y \leqslant x\}.$$

[5] available at `http://homepages.laas.fr/mmjoldes/campary/`

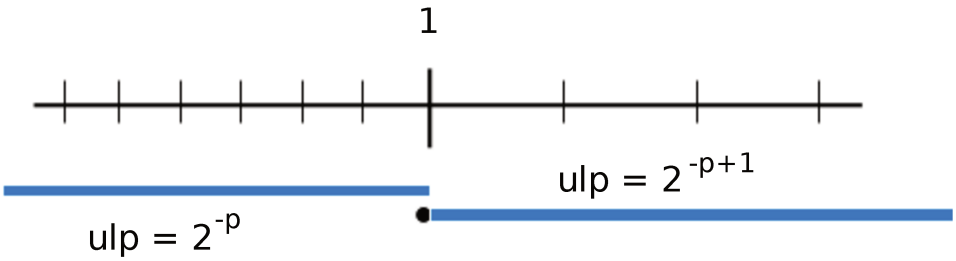


Figure 1.3: Floating-point discretization of the real line.

- Round towards $+\infty$: $\mathbf{RU}(x)$ is the smallest floating-point number greater or equal to x, possibly equal to the exceptional value $+\infty$:

$$\mathbf{RU}(x) = \min\{y \in \mathbb{F}, y \geqslant x\}.$$

- Round towards 0: $\mathbf{RZ}(x)$ is the closest floating-point number to x of absolute value smaller or equal to $|x|$:

$$\mathbf{RZ}(x) = \begin{cases} \mathbf{RU}(x) & \text{if } x \leqslant 0, \\ \mathbf{RD}(x) & \text{if } x \geqslant 0. \end{cases}$$

- Round to nearest: $\mathbf{RN}(x)$ is the closest floating-point number to x. If x is the exact middle point of two consecutive floating-point numbers, then the most commonly used tie-breaking rule is to choose among these two numbers the one with an even integral significand:

$$\mathbf{RN}(x) = \begin{cases} x & \text{if } x \in \mathbb{F}, \\ \mathbf{RD}(x) & \text{if } x < (\mathbf{RD}(x) + \mathbf{RU}(x))/2 \\ & \text{or } x = (\mathbf{RD}(x) + \mathbf{RU}(x))/2 \text{ and } \mathbf{RD}(x) \text{ has an even significand,} \\ \mathbf{RU}(x) & \text{if } x > (\mathbf{RD}(x) + \mathbf{RU}(x))/2 \\ & \text{or } x = (\mathbf{RD}(x) + \mathbf{RU}(x))/2 \text{ and } \mathbf{RU}(x) \text{ has an even significand.} \end{cases}$$

  Note that another tie-breaking rule (ties away from 0) is also defined by the IEEE 754 standard.

Proposition 1.8 (Rounding errors) *For a nonzero real number x with $|x| \leqslant 2^{e_{\max}+1} - 2^{e_{\max}-p+1}$ (the largest representable floating-point number), we have the following bounds for rounding errors:*

$$|x - \mathbf{RD}(x)| < \operatorname{ulp}(x), \qquad |x - \mathbf{RU}(x)| < \operatorname{ulp}(x),$$
$$|x - \mathbf{RZ}(x)| < \operatorname{ulp}(x), \qquad |x - \mathbf{RN}(x)| \leqslant \frac{1}{2}\operatorname{ulp}(x).$$

Arithmetic operations

A challenging property of floating-point numbers is that $\mathbb{F}$ is not closed under arithmetic operations: for $x, y \in \mathbb{F}$, $x+y$, $x-y$, xy, x/y and $\sqrt{x}$ are not exactly representable by floating-point numbers in the same format $\mathbb{F}$ in general. Therefore, arithmetic operations only admit

approximate counterparts in $\mathbb{F}$. At the beginning of floating-point arithmetic, the specifications of hardware implementations were rather vague concerning those approximated operations. In practice, the same computation performed on different machines could produce quite different results. Since the adoption of the IEEE 754-1985 standard by most architectures, things have improved considerably. A major requirement of the standard is for these basic operations to be correctly rounded with respect to the four rounding modes of **Proposition 1.8** [113]: for any operation $* \in \{+, -, \times, /, \sqrt{}\}$ and rounding mode $\circ \in \{\mathsf{RU}, \mathsf{RD}, \mathsf{RZ}, \mathsf{RN}\}$, there is a corresponding floating-point operation $\widetilde{*}^{\circ}$ satisfying

$$x \,\widetilde{*}^{\circ} y = \circ(x * y), \qquad \text{for all } x, y \in \mathbb{F}.$$

By default, standard processors use the round-to-nearest mode. The C file `fenv.h` defines macros `FE_DOWNWARD`, `FE_UPWARD`, `FE_TOWARDZERO`, and `FE_TONEAREST` for, respectively, **RD**, **RU**, **RZ**, and **RN**. Functions `int fegetround(void)` and `int fesetround(int round)` allow one to get the current rounding mode and set it to another one. However, changing the rounding mode has a non-negligible cost and should be used with caution.

In the MPFR software library [83], the rounding mode (`MPFR_RNDD`, `MPFR_RNDU`, `MPFR_RNDZ`, or `MPFR_RNDN`) has to be specified for each elementary operation. The result is guaranteed to be correctly rounded, whatever the precision p is.

Remark 1.9 *The directed rounding modes* **RD** *and* **RU** *will be particularly useful for interval arithmetics in Section 1.3. For convenience, they will be denoted by* $\triangledown(x)$ *and* $\triangle(x)$*, instead of* $\mathsf{RD}(x)$ *and* $\mathsf{RU}(x)$*.*

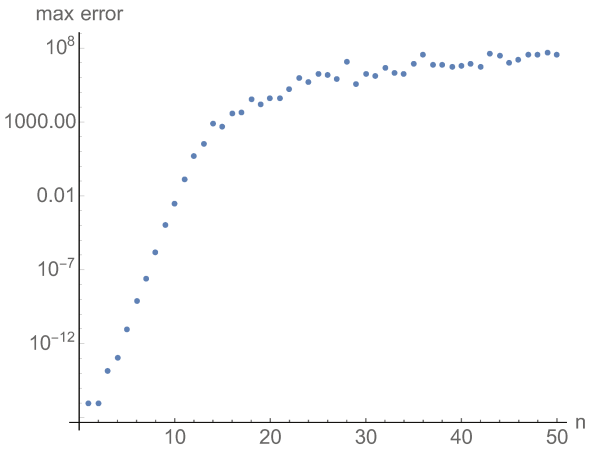
Example 1.10 (Hilbert matrix inversion using floating-point arithmetic) *Pursuing Example 1.3, we now address the inversion of* H_n*, the Hilbert matrix, using a standard Gaussian elimination procedure (with pivots) with double-precision machine floating-point numbers. The first remark concerns efficiency: the computation time is drastically reduced compared to the same experiment with rational numbers, and the timings are now in accordance with a time complexity in* $\mathcal{O}(n^3)$ *(see Table 1.4a).*

However, the Hilbert matrix is well-known to be particularly ill-conditioned *(see* [17] *and references therein), meaning that the propagation and accumulation of small elementary rounding errors due to floating-point arithmetic may lead to significant errors in the end. Figure 1.4b plots the maximum absolute value of the entries in the floating-point evaluation of the matrix* $H_n^{-1} H_n - 1$*.*

Rounding error analysis is an important topic of numerical analysis. It consists in providing bounds on rounding errors in the final result of more or less complex algorithms, such as dot products, polynomial evaluation schemes, linear algebra routines, etc. [106]. Although we do not give details here, some insight will be provided in **Chapter 8**, regarding a method for obtaining polynomials which minimize both approximation and evaluation errors.

n	t (ms)	n	t (ms)
		260	102
20	1	280	135
40	1	300	159
60	1	320	187
80	4	340	226
100	7	360	267
120	13	380	312
140	22	400	365
160	28	420	424
180	35	440	490
200	45	460	558
220	62	480	669
240	80	500	751

(a) Execution time of a Gaussian inversion procedure applied to H_n, using machine double-precision floating-point numbers.



(b) Maximum absolute error in the entries of $H_n^{-1}H_n - 1$.

1.3 Interval arithmetic

While in many cases, floating-point arithmetic offers an acceptable trade-off between efficiency and accuracy, some situations (like inverting Hilbert matrices in **Example 1.10**) require a particular care due to potentially disastrous numerical errors. Numerical analysis provides a wide range of algorithms with stronger stability properties in such situations **[106]**, yet some applications need *rigorous tools*, which offer guarantees concerning the result at mathematical level.

Invented at the end of the 50s independently by Sunaga **[238]** and Moore **[173]** (see **[221,** Sec. 1.5**]**), then significantly developed in the 70s-80s by Kulisch **[143]** and colleagues, interval arithmetic is an essential building block of rigorous numerics. The key idea consists in using real intervals with representable endpoints (e.g., floating-point numbers) as rigorous enclosures of real numbers, and providing operations which preserve correctness. For example, from $\pi \in [3.1415, 3.1416]$ and $e \in [2.7182, 2.7183]$, one obtains:

$$\pi + e \in [3.1415, 3.1416] \oplus [2.7182, 2.7183] = [5.8597, 5.8599].$$

Precise definitions of such an arithmetic are given in **Section 1.3.1**. We then look into several rigorous techniques based on interval arithmetic, such as *interval subdivision* (**Section 1.3.2**) and *interval Newton's method* (**Section 1.3.3**).

Contrary to exact representations discussed in **Section 1.1**, an interval does not denote a single value in general, but a set of possible values. Moreover, there is most of the time no guarantee concerning the width of the intervals. In particular, interval arithmetic does not claim to be more accurate than floating-point arithmetic – but it is *safe*. In that regard, **Section 1.3.4** gives an insight of situations where interval arithmetic fails to compute reasonably tight enclosures. While that led part of the community to neglect this topic, a wiser standpoint is to consider interval arithmetic not as a panacea – substitute all FP numbers by intervals in your programs! – but as a low-level building block widely used in rigorous numerics, for

example in the *rigorous polynomial approximations* of **Chapter 3**.

This chapter is a rather compact, yet sufficient introduction to interval analysis for the purpose of this manuscript. We refer to the following reference textbooks **[174, 247, 221, 183]** for a more detailed account of this topic, as well as this dedicated webpage[6].

1.3.1 ▶ Intervals: definitions, operations and properties

In the following, we define the set of intervals with endpoints in $\mathbb{B}$ for some subset $\mathbb{B}$ of $\mathbb{R}$. Usually, $\mathbb{B} = \mathbb{R}$ when considering mathematical real intervals, whereas $\mathbb{B} = \mathbb{F}$ (for some set $\mathbb{F}$ of floating-point numbers defined in **Section 1.2**) will be used for intervals with floating-point endpoints, which are concrete computational objects.

Definition 1.11 (Intervals) *For $\mathbb{B} \subseteq \mathbb{R}$, the set $\mathbb{I}_\mathbb{B}$ of closed intervals of the real line with endpoints in $\mathbb{B}$ is*

$$\mathbb{I}_\mathbb{B} := \{\mathbb{x} = [\underline{x}, \overline{x}] = \{x \in \mathbb{R} \mid \underline{x} \leqslant x \leqslant \overline{x}\}, \underline{x} \in \mathbb{B} \cup \{-\infty\} \ \wedge \ \overline{x} \in \mathbb{B} \cup \{+\infty\}\},$$

where, by convention, the endpoints $\underline{x}$ and $\overline{x}$ of $\mathbb{x}$ are excluded if equal to $\pm\infty$.

We call $\underline{x}$ (resp. $\overline{x}$) the lower bound *(resp.* upper bound*) of $\mathbb{x}$, $w(\mathbb{x}) := \overline{x} - \underline{x}$ its* width*, and* $\mathrm{mag}(\mathbb{x}) := \max(\overline{x}, -\underline{x})$ *its* magnitude *(both equal to $+\infty$ if $\underline{x}$ or $\overline{x}$ is $\pm\infty$). Note that $[\underline{x}, \overline{x}] = \varnothing$ if $\overline{x} < \underline{x}$.*

Remark 1.12 *It may be convenient to add an extra constant $\text{⫫}_\mathbb{B}$ to the set $\mathbb{I}_\mathbb{B}$ that corresponds to a possible execution error during the computation. In that setting, $\text{⫫}_\mathbb{B}$ is the largest element of $\mathbb{I}_\mathbb{B}$ (for the inclusion): it contains all the real numbers, plus possibly some error symbols.*

Definition 1.13 (Point intervals) *For any $x \in \mathbb{R}$, the singleton $[x] := [x, x] = \{x\}$ is called a* point interval*. This allows for injecting $\mathbb{R}$ into $\mathbb{I}_\mathbb{R}$.*

Since $\mathbb{F}$ is discrete in $\mathbb{R}$, one can only define thin intervals *mapping $\mathbb{R}$ to $\mathbb{I}_\mathbb{F}$:*

$$[x]_\mathbb{F} := \begin{cases} \{x\} & \text{if } x \in \mathbb{F}, \\ [\triangledown(x), \triangle(x)] & \text{otherwise.} \end{cases}$$

Note that two real numbers have the same corresponding thin interval if they both lie between the same two consecutive floating-point numbers.

Definition 1.14 (Set-theoretic operations) *The intersection of two intervals is an interval:*

$$\mathbb{x} \mathbin{⩀} \mathbb{y} \quad := \quad \begin{cases} \mathbb{y} & \text{if } \mathbb{x} = \text{⫫}_\mathbb{B}, \\ \mathbb{x} & \text{if } \mathbb{y} = \text{⫫}_\mathbb{B}, \\ [\max(\underline{x}, \underline{y}), \min(\overline{x}, \overline{y})], & \text{if } \mathbb{x} = [\underline{x}, \overline{x}] \text{ and } \mathbb{y} = [\underline{y}, \overline{y}]. \end{cases}$$

[6] `http://www.cs.utep.edu/interval-comp/`

The union of two intervals is not necessarily an interval, but it is always contained in an interval called the convex hull*:*

$$\mathbb{x} \bigcirc\!\!\!\!\vee \mathbb{y} \quad := \quad \begin{cases} \perp\!\!\!\perp_{\mathbb{B}} & \text{if } \mathbb{x} = \perp\!\!\!\perp_{\mathbb{B}} \text{ or } \mathbb{y} = \perp\!\!\!\perp_{\mathbb{B}}, \\ \mathbb{y} & \text{if } \mathbb{x} = \varnothing, \\ \mathbb{x} & \text{if } \mathbb{y} = \varnothing, \\ [\min(\underline{x}, \underline{y}), \max(\overline{x}, \overline{y})], & \text{if } \mathbb{x} = [\underline{x}, \overline{x}] \text{ and } \mathbb{y} = [\underline{y}, \overline{y}]. \end{cases}$$

Remark 1.15 (Ball arithmetic **[252]**) *As an alternative to interval arithmetic, one can also use a* midpoint-radius *representation, called* ball arithmetic*, to define intervals or real numbers. Although completely equivalent from the mathematical point of view, this approach has advantages and drawbacks over interval arithmetic when implemented with floating-point numbers* **[252, 219, 125]**.

Interval extension: a fundamental requirement

The notion of *interval extension*, or *inclusion principle*, is central in interval arithmetic **[221,** Sec. 5.5**]**.

Definition 1.16 (Interval extension and range) *Let $k \geqslant 1$, $A \subseteq \mathbb{R}^k$ and $f : A \to \mathbb{R}$ be a k-ary function. An interval function $\mathbb{f} : {\mathbb{I}_{\mathbb{B}}}^k \to \mathbb{I}_{\mathbb{B}}$ is called an ($\mathbb{I}_{\mathbb{B}}$-)*interval extension *of f if*

$$\forall \mathbb{x}_1, \ldots, \mathbb{x}_k \in \mathbb{I}_{\mathbb{B}},\ \forall x_1, \ldots, x_k \in \mathbb{R},$$
$$\mathbb{x}_1 \times \ldots \mathbb{x}_k \subseteq A \quad \wedge \quad \forall i \in [\![1, k]\!],\ (x_i \in \mathbb{x}_i) \ \Rightarrow\ f(x_1, \ldots, x_k) \in \mathbb{f}(\mathbb{x}_1, \ldots, \mathbb{x}_k).$$

If A is compact and f is continuous over A, then the range *of f over $\mathbb{x}_1 \times \cdots \times \mathbb{x}_k \subseteq A$, defined as*

$$f(\mathbb{x}_1, \ldots, \mathbb{x}_k) := \{f(x_1, \ldots, x_k), x_i \in \mathbb{x}_i \text{ for } i \in [\![1, k]\!]\},$$

gives the tightest $\mathbb{I}_{\mathbb{R}}$-interval extension of f. In the general case, the range may not be a closed interval, but it is still contained in all $\mathbb{I}_{\mathbb{R}}$-interval extensions of f.

Remark 1.17 *When $\mathbb{I}_{\mathbb{B}}$ is defined to contain an exceptional interval $\perp\!\!\!\perp_{\mathbb{B}}$, **Definition 1.16** can be extended by requiring that $\mathbb{f}(\mathbb{x}_1, \ldots, \mathbb{x}_k) = \perp\!\!\!\perp_{\mathbb{B}}$ if $\mathbb{x}_1 \times \cdots \times \mathbb{x}_k \nsubseteq A$, that is one of the arguments of f potentially lies outside its domain of definition.*

In addition, the *isotonicity* property requires that more precise input gives more precise output, in terms of interval inclusion **[174,** Sec. 4.3**]**.

Definition 1.18 (Inclusion isotonicity) *An interval extension $\mathbb{f}$ of f is said* inclusion isotonic *if*

$$\forall \mathbb{x}_1, \ldots, \mathbb{x}_k, \mathbb{y}_1, \ldots, \mathbb{y}_k \in \mathbb{I}_{\mathbb{B}},\ \forall i \in [\![1, k]\!],\ \mathbb{x}_i \subseteq \mathbb{y}_i \Rightarrow \mathbb{f}(\mathbb{x}_1, \ldots, \mathbb{x}_k) \subseteq \mathbb{f}(\mathbb{y}_1, \ldots, \mathbb{y}_k).$$

Interval arithmetic operations

The range of arithmetic operations on real numbers, explicitly given in the following definition, yields a natural isotonic $\mathbb{I}_{\mathbb{R}}$-interval extension $\circledast$ for each elementary operation $*$ **[174,** Sec. 2.3**]**.

Definition 1.19 (Arithmetic operations on real intervals) *Arithmetic operations on $\mathbb{I}_{\mathbb{R}}$ are defined as the range of the corresponding operations in $\mathbb{R}$. For any $\mathbb{x} = [\underline{x}, \overline{x}]$ and $\mathbb{y} = [\underline{y}, \overline{y}]$ in $\mathbb{I}_{\mathbb{R}}$,*

$$
\begin{aligned}
\mathbb{0} &:= [0], \qquad \mathbb{1} := [1], \\
\mathbb{x} \oplus \mathbb{y} &:= [\underline{x} + \underline{y}, \overline{x} + \overline{y}], \\
\ominus \mathbb{y} &:= [-\overline{y}, -\underline{y}], \\
\mathbb{x} \ominus \mathbb{y} &:= [\underline{x} - \overline{y}, \overline{x} - \underline{y}], \\
\mathbb{x} \otimes \mathbb{y} &:= [\min(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}), \max(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y})], \\
\mathbb{y}^{-①} &:= \begin{cases} \perp\!\!\!\perp_{\overline{\mathbb{R}}} & \text{if } 0 \in \mathbb{y}, \\ [\min(\underline{y}^{-1}, \overline{y}^{-1}), \max(\underline{y}^{-1}, \overline{y}^{-1})] & \text{otherwise}, \end{cases} \\
\mathbb{x} \oslash \mathbb{y} &:= \mathbb{x} \otimes \mathbb{y}^{-①}, \\
\mathbb{x}^{②} &:= \begin{cases} [0, \max(\underline{x}^2, \overline{x}^2)] & \text{if } 0 \in \mathbb{x}, \\ [\min(\underline{x}^2, \overline{x}^2), \max(\underline{x}^2, \overline{x}^2)] & \text{otherwise}. \end{cases}
\end{aligned}
$$

Note that, in the definition of $\otimes$, 0 is absorbent over $\pm\infty$. Moreover, $\perp\!\!\!\perp_{\overline{\mathbb{R}}}$ is absorbent for all these operations.

It is important to notice that, if the operations given above are correct w.r.t. the inclusion principle, they do not define a ring nor field structure over $\mathbb{I}_{\mathbb{R}}$. Indeed, as is shown by the proposition below (whose proof may be found in **[129,** Prop. 1.3.11**]**), only weaker properties are satisfied.

Proposition 1.20 *Interval arithmetic partially preserves the field structure of $\mathbb{R}$:*

- *Associativity and commutativity of addition are preserved:*

$$(\mathbb{x} \oplus \mathbb{y}) \oplus \mathbb{z} = \mathbb{x} \oplus (\mathbb{y} \oplus \mathbb{z}), \qquad \mathbb{x} \oplus \mathbb{y} = \mathbb{y} \oplus \mathbb{x}.$$

- *$\mathbb{0}$ is neutral for addition: $\mathbb{x} \oplus \mathbb{0} = \mathbb{x}$.*
- *$\mathbb{0} \subseteq \mathbb{x} \ominus \mathbb{x}$, but equality does not hold in general.*
- *Associativity and commutativity of multiplication are preserved:*

$$(\mathbb{x} \otimes \mathbb{y}) \otimes \mathbb{z} = \mathbb{x} \otimes (\mathbb{y} \otimes \mathbb{z}), \qquad \mathbb{x} \otimes \mathbb{y} = \mathbb{y} \otimes \mathbb{x}.$$

- *$\mathbb{1}$ is neutral for multiplication: $\mathbb{1} \otimes \mathbb{x} = \mathbb{x}$.*
- *$\mathbb{0}$ is absorbent for multiplication: $\mathbb{0} \otimes \mathbb{x} = \mathbb{0}$ whenever $\mathbb{x} \neq \perp\!\!\!\perp_{\overline{\mathbb{R}}}$.*
- *$\mathbb{1} \subseteq \mathbb{x} \oslash \mathbb{x}$, but equality does not hold in general.*
- *Multiplication is only* sub-distributive *over addition:*

$$\mathbb{x} \otimes (\mathbb{y} \oplus \mathbb{z}) \subseteq \mathbb{x} \otimes \mathbb{y} \oplus \mathbb{x} \otimes \mathbb{z}.$$

- *$\mathbb{x}^{②} \subseteq \mathbb{x} \otimes \mathbb{x}$, but equality does not hold in general.*

Floating-point interval arithmetic

On the computational side, arithmetic operations on floating-point intervals are defined similarly but with directed rounding, in order to maintain floating-point endpoints. Although those operations do not coincide anymore with the exact range, they are still valid interval extensions of the corresponding operations in $\mathbb{R}$.

Definition 1.21 (Arithmetic operations on floating-point intervals) *Arithmetic operations on $\mathbb{I}_\mathbb{F}$ are defined as follows. For any $\mathbb{x} = [\underline{x}, \overline{x}]$ and $\mathbb{y} = [\underline{y}, \overline{y}]$ in $\mathbb{I}_\mathbb{F}$,*

$$
\begin{aligned}
\mathbb{x} \mathbin{\widetilde{\oplus}} \mathbb{y} &:= [\nabla(\underline{x} + \underline{y}), \triangle(\overline{x} + \overline{y})], \\
\widetilde{\ominus}\ \mathbb{y} &:= [-\overline{y}, -\underline{y}], \\
\mathbb{x} \mathbin{\widetilde{\ominus}} \mathbb{y} &:= [\nabla(\underline{x} - \overline{y}), \triangle(\overline{x} - \underline{y})], \\
\mathbb{x} \mathbin{\widetilde{\otimes}} \mathbb{y} &:= [\min(\nabla(\underline{x}\underline{y}), \nabla(\underline{x}\overline{y}), \nabla(\overline{x}\underline{y}), \nabla(\overline{x}\underline{y})), \max(\triangle(\underline{x}\underline{y}), \triangle(\underline{x}\overline{y}), \triangle(\overline{x}\underline{y}), \triangle(\overline{x}\underline{y}))], \\
y^{\widetilde{-1}} &:= \begin{cases} \mathbb{\perp}_\mathbb{F} & \text{if } 0 \in \mathbb{y}, \\ [\min(\nabla(\underline{y}^{-1}), \nabla(\overline{y}^{-1})), \max(\triangle(\underline{y}^{-1}), \triangle(\overline{y}^{-1}))] & \text{otherwise,} \end{cases} \\
\mathbb{x} \mathbin{\widetilde{\oslash}} \mathbb{y} &:= \begin{cases} \mathbb{\perp}_\mathbb{F} \quad \text{if } 0 \in \mathbb{y}, \quad \text{otherwise:} \\ [\min(\nabla(\underline{x}/\underline{y}), \nabla(\underline{x}/\overline{y}), \nabla(\overline{x}/\underline{y}), \nabla(\overline{x}/\underline{x})), \max(\triangle(\underline{x}/\underline{y}), \triangle(\underline{x}/\overline{y}), \triangle(\overline{x}/\underline{y}), \triangle(\overline{x}/\underline{y}))], \end{cases} \\
\mathbb{x}^{\widetilde{2}} &:= \begin{cases} [0, \max(\triangle(\underline{x}^2), \triangle(\overline{x}^2))] & \text{if } 0 \in \mathbb{x}, \\ [\min(\nabla(\underline{x}^2), \nabla(\overline{x}^2)), \max(\triangle(\underline{x}^2), \triangle(\overline{x}^2))] & \text{otherwise.} \end{cases}
\end{aligned}
$$

Implementations of FP interval arithmetic are an essential component of rigorous numerics software. Beyond their correctness, they are also optimized to reduce as much as possible the efficiency gap between floating-point and interval arithmetics. It is therefore highly recommended to resort to such implementations if possible, rather than writing ad-hoc code, which is often error-prone and most of the time less competitive. Reference libraries include: MPFI [210], in C and based on MPFR [83], with a C++ wrapper; C-XSC [139], in C++; INTLAB [220], an extension of MATLAB; ARB [125], a very efficient implementation of ball arithmetic in C (see **Remark 1.15**). A detailed survey comparing these implementations may be found in [100].

On the formal proof side, CoqInterval [171] provides a formalization of interval arithmetic inside the Coq proof assistant, based on emulated floating-point numbers from the Flocq library [30].

Natural interval extensions using interval arithmetic

Interval extensions for elementary functions are built based on their monotonicity properties. We assume that computable isotonic interval extensions $\widetilde{\sqrt{}}$, $\widetilde{\exp}$, $\widetilde{\ln}$, $\widetilde{\cos}$, $\widetilde{\sin}$, $\widetilde{\tan}$, $\widetilde{\arccos}$, $\widetilde{\arcsin}$ and $\widetilde{\arctan}$ are available for $\sqrt{}$, exp, ln, cos, sin, tan, acos, asin and atan. This is for example the case in the MPFI library.

Given a k-ary function $f : A \to \mathbb{R}$ with $A \subseteq \mathbb{R}^k$, defined using elementary functions, arithmetic operations and composition, one can construct a so-called *natural interval extension* $\mathbb{f}$ of f by replacing all the symbols of functions and operators by the corresponding interval ones. Such a $\mathbb{f}$ is a valid interval extension for f, as stated by the following theorem (see [247, Thm. 3.8]).

Theorem 1.22 (Fundamental theorem of interval arithmetic) *The natural interval extension $\mathbb{f}$ of a k-ary function $f : A \to \mathbb{R}$ with $A \subseteq \mathbb{R}^k$ is an isotonic interval extension of f, that is, $\mathbb{f}$ satisfies the inclusion and isotonicity principles w.r.t. f.*

Natural interval extensions are sometimes called *non-intrusive* methods, since a piece of code written with floating-point operations can easily be transposed to interval arithmetic by overloading the operators and elementary functions with intervals, thus not modifying the code itself.

1.3.2 ▶ Interval subdivision techniques

Let $f : \mathbb{i} \to \mathbb{R}$ be a continuous function defined over a compact interval $\mathbb{i}$, and $\mathbb{f}$ be an interval extension of it. In general, $\mathbb{f}(\mathbb{i})$ only provides a rough overapproximation of the exact range $f(\mathbb{i}) = [\min_{x \in \mathbb{i}} f(x), \max_{x \in \mathbb{i}} f(x)]$. This may be quite inconvenient for various tasks, such as positivity check or global optimization.

A common way to address such problems is to use *interval subdivision*, possibly combined with *branch and bound* techniques (see **[247,** Chap. 5**]**. The idea is the following:

- The interval $\mathbb{i}$ is subdivided into smaller intervals $\mathbb{j}_k \subseteq \mathbb{i}$.
- Each time $\mathbb{f}(\mathbb{j}_k)$ is considered not precise enough, $\mathbb{j}_k$ is further subdivided into smaller intervals.
- When $\mathbb{f}(\mathbb{j}_k)$ satisfies a *pruning condition* (meaning that it contains enough information for the target problem), $\mathbb{j}_k$ is not subdivided anymore.
- A maximum subdivision depth is fixed by the user, so that the process eventually terminates.

Let us illustrate this process with two examples.

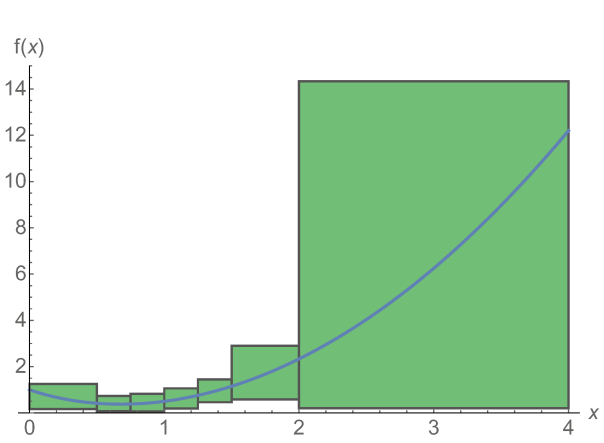
Example 1.23 (Checking positivity) *Consider the function $f : x \mapsto \frac{1}{1+x} - x + x^2$, and prove that $f(x) > 0$ for $x \in [0, 4]$ (see **Figure 1.5a**), using only interval analysis techniques – being clear that classical analysis techniques easily handle this toy example.*

Let $\mathbb{f}$ denote the natural interval extension of f: the first attempt consists in applying it to $[0, 4]$:

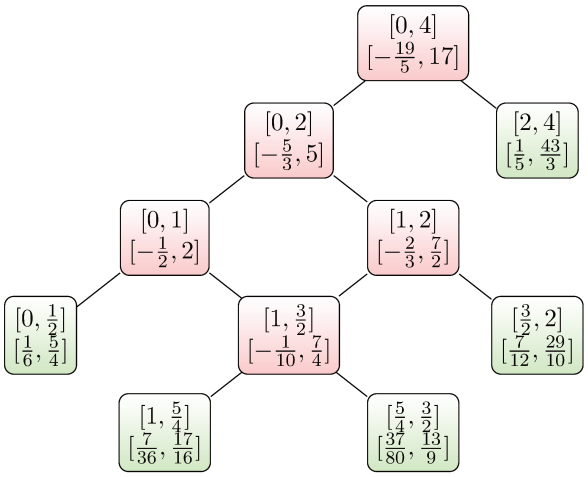
$$\mathbb{f}([0,4]) = \mathbb{1} \oslash (\mathbb{1} \oplus [0,4]) \ominus [0,4] \oplus [0,4]^{②}$$
$$= \mathbb{1} \oslash [1,5] \ominus [0,4] \oplus [0,16] = [\frac{1}{5}, 1] \ominus [0,4] \oplus [0,16] = [-\frac{19}{5}, 17].$$

Unfortunately, the range enclosure $[-\frac{19}{5}, 17]$ is not sufficient to assert the positivity of f over $[0, 4]$. Therefore, $[0, 4]$ is subdivided *into (for example) $[0, 2]$ and $[2, 4]$, and $\mathbb{f}$ is evaluated on both intervals. Since $\mathbb{f}([2, 4]) = [\frac{1}{5}, \frac{43}{3}]$ guarantees the positivity over $[2, 4]$, this interval does not need subdividing anymore: this is the* pruning condition*. On the contrary, $\mathbb{f}([0, 2]) = [-\frac{5}{3}, 5]$ is not precise enough to complete the proof, thereby requiring further subdividing.*

*Finally, a subdivision depth at least equal to 4 is needed to prove the positivity of f over $[0, 4]$. The subdivision tree and the resulting plot are depicted in **Figure 1.5**.*



(a) Range enclosure associated to an interval subdivision, proving the positivity of f over $[0, 4]$.



(b) Subdivision tree: the leaves (in green) exhibit positive ranges.

■ **Figure 1.5**: Interval subdivision technique proving the positivity of $f : x \mapsto \frac{1}{1+x} - x + x^2$ over $[0, 4]$.

■ **Example 1.24** (Global optimization) *Let f be a continuous function over a compact interval $\mathbb{i}$, for which we have an isotonic interval extension $\mathbb{f}$. Our goal is to find the global maximum $m = \max_{x \in \mathbb{x}} f(x)$ over $\mathbb{i}$, using interval analysis techniques. Here we describe a very naive optimization procedure that returns a valid enclosure $\mathbb{m}$ for m. Due to its mathematical simplicity, such a strategy is particularly suitable for certified global optimizers, e.g., the* `interval` *tactic in the* Coq *library* CoqInterval **[171]**.

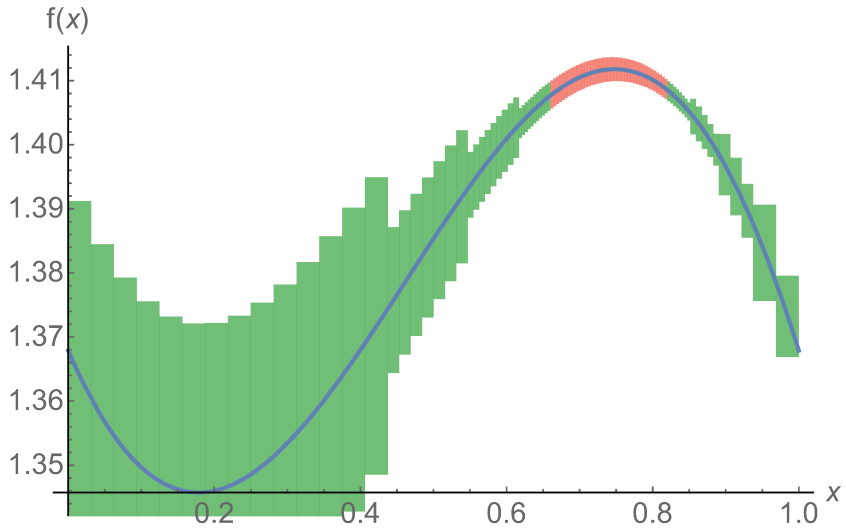
At each step ℓ, the algorithm maintains an interval subdivision $\mathbb{J}_\ell = \{\mathbb{j}_{\ell k}, k \in [\![0, n_\ell]\!]\}$ of $\mathbb{i}$ together with a valid interval enclosure $\mathbb{m}_\ell$ of m:

- *The process is initialized with $\mathbb{J}_0 = \{\mathbb{i}\}$ and $\mathbb{m}_0 = \mathbb{f}(\mathbb{i})$.*
- *At step ℓ, having $\mathbb{J}_\ell$ and $\mathbb{m}_\ell = [\underline{m}_\ell, \overline{m}_\ell]$, the algorithm sorts the $\mathbb{j}_{\ell k}$ into two categories: If $\mathbb{f}(\mathbb{j}_{\ell k}) = [\underline{y}_{\ell k}, \overline{y}_{\ell k}]$ and $\overline{y}_{\ell k} < \underline{m}_\ell$, then it is clear that no point of $\mathbb{j}_{\ell k}$ realizes the maximum m. Therefore, this interval is simply discarded. Otherwise, $\mathbb{y}_{\ell k}$ may contain a point x such that $f(x) = m$, and the algorithm decides to subdivide it.*
- *At the end of each step, the new interval subdivision $\mathbb{J}_{\ell+1}$ is obtained from $\mathbb{J}_\ell$ by replacing all the $\mathbb{j}_{\ell k}$ that were subdivided by the corresponding subintervals. Moreover, $\mathbb{m}_{\ell+1} := [\underline{m}_{\ell+1}, \overline{m}_{\ell+1}]$, where:*

$$\underline{m}_{\ell+1} := \max_{1 \leqslant k \leqslant n_{\ell+1}} \underline{y}_{(\ell+1)k}, \qquad \overline{m}_{\ell+1} := \max_{1 \leqslant k \leqslant n_{\ell+1}} \overline{y}_{(\ell+1)k}.$$

- *The process stops when a specified accuracy for $\mathbb{m}_\ell$ has been obtained, or a given maximum subdivision depth has been reached.*

Figure 1.6 *depicts the result of this procedure, for $f(x) = e^x - e^{-(x-1)^2}$ over $[0, 1]$, with a specified accuracy $\varepsilon = 0.005$. We get $\mathbb{m} = [1.40994, 1.41364]$. The boxes on the plot represent the interval subdivision at the last step, evaluated using $\mathbb{f}$. The green ones are those for which*



■ **Figure 1.6**: Rigorous global optimization using interval subdivision, for $f : x \mapsto e^x - e^{-(x-1)^2}$ over $[0, 1]$. Green subintervals cannot contain the maximum, red ones need further subdivision.

the upper bound is smaller than $\underline{m}$*, so that we are sure that they do not contain the optimal solution. The red ones may contain it. In the end, this proves that any* x *satisfying* $f(x) = m$ *is inside the union of the red intervals, that is* $x \in [\frac{169}{256}, \frac{105}{128}]$.

1.3.3 ▶ Interval Newton's method

Newton's method is a classical tool in numerical analysis to approximate roots of sufficiently smooth functions with a quadratic asymptotic rate of convergence **[192,** Chap. 7**]**. Interval analysis offers a rigorous counterpart to it, called *Interval Newton Method*. We provide the essential ingredients of this method, and refer the reader to **[173]**, **[174,** Chap. 8**]** and **[247,** Sec. 5.1.2**]** for a more thorough survey on that topic.

Let $\mathbb{i}$ be a compact interval of $\mathbb{R}$ and $f : \mathbb{i} \to \mathbb{R}$ be a function of class $\mathcal{C}^1$ having at least one root x^* in $\mathbb{i}$. For any $x \in \mathbb{i}$, the mean value theorem guarantees the existence of $\xi \in \mathbb{i}$ such that

$$f(x^*) = f(x) + f'(\xi)(x^* - x).$$

Since $f(x^*) = 0$, this gives:

$$\text{either} \quad f(x) = 0, \qquad \text{or} \quad x^* = x - \frac{f(x)}{f'(\xi)}. \tag{1.2}$$

This motivates the definition of the interval Newton iteration.

■ **Definition 1.25** *Suppose that interval extensions* $\mathbb{f}$ *and* $\mathbb{df}$ *are available for* f *and* f'*. Then the Interval Newton's method computes a sequence* $(\mathbb{x}_k)_{k \geqslant 0}$ *of intervals, defined by:*

- $\mathbb{x}_0 \subseteq \mathbb{i}$ *is any initial interval enclosure of* x^* *(possibly* $\mathbb{i}$ *itself).*

- *At step k, the next interval $\mathbb{x}_{k+1}$ is computed as*

$$\mathbb{x}_{k+1} = \mathbb{x}_k \oslash N(\mathbb{x}_k), \quad \textit{where} \quad N(\mathbb{x}) = [\mathrm{mid}(\mathbb{x})] \ominus \mathbb{f}([\mathrm{mid}(\mathbb{x})]) \oslash \mathbb{df}(\mathbb{x}).$$

Theorem 1.26 (Correctness of Interval Newton's method **[174,** Thm. 8.1**]**) *If $x^* \in \mathbb{x}_0 \subseteq \mathbb{i}$ is a root of f, then $x^* \in \mathbb{x}_k$ for all $k \geqslant 0$.*

Proof. The assertion is proved by induction on k. The base case $k = 0$ is the assumption of the theorem. For the inductive step, one just needs to prove that $x^* \in \mathbb{x}$ implies $x^* \in N(\mathbb{x})$, for any $\mathbb{x} \subseteq \mathbb{i}$. According to Equation **(1.2)**, where x is instantiated with $\tilde{x} = \mathrm{mid}(\mathbb{x})$, two cases are possible:

- Either $f(\tilde{x}) = 0$, in which case $\tilde{x} \in N(\mathbb{x})$ because $0 \in \mathbb{f}([\tilde{x}])$. If $\tilde{x} = x^*$, the proof is completed. Otherwise, by the mean value theorem, $0 \in \mathbb{df}(\mathbb{x})$, hence $N(\mathbb{x}) = \perp\!\!\!\perp$.
- Or there exists a $\xi \in [\min(\tilde{x}, x^*), \max(\tilde{x}, x^*)] \subseteq \mathbb{x}$ such that $x^* = \tilde{x} - \frac{f(\tilde{x})}{f'(\xi)}$. Then $\tilde{x} \in N(\mathbb{x})$ follows from the correctness of interval operations.

□

Remark 1.27 *If f' vanishes over $\mathbb{x}_0$, then $\mathbb{x}_k = \mathbb{x}_0$ for all $k \geqslant 1$ (since $N(\mathbb{x}_0) = \perp\!\!\!\perp$). Otherwise, f is strictly monotonic over $\mathbb{x}_0$ and therefore it changes sign once, with x^* its unique root. In that case, Interval Newton's method can be combined with dichotomy by modifying slightly its definition:*

$$\mathbb{y}_k = \begin{cases} [\underline{x}_k, \mathrm{mid}(\mathbb{x}_k)] & \textit{if } \mathbb{f}([\underline{x}_k]) \otimes \mathbb{f}([\mathrm{mid}(\mathbb{x}_k)]) \subseteq [-\infty, 0], \\ [\mathrm{mid}(\mathbb{x}_k), \overline{x}_k] & \textit{if } \mathbb{f}([\underline{x}_k]) \otimes \mathbb{f}([\mathrm{mid}(\mathbb{x}_k)]) \subseteq [0, +\infty], \\ \mathbb{x}_k & \textit{otherwise}, \end{cases}$$

$$\mathbb{x}_{k+1} := \mathbb{y}_k \oslash ([\mathrm{mid}(\mathbb{x}_k)] \ominus \mathbb{f}([\mathrm{mid}(\mathbb{x}_k)]) \oslash \mathbb{df}(\mathbb{y}_k)) \,.$$

Analogously to the classical Newton's method, one can prove a quadratic asymptotic convergence rate for this interval version, at least for restricted classes of function f. This means that asymptotically, each iteration *doubles* the number of correct digits.

Theorem 1.28 (Quadratic convergence rate of Interval Newton's method **[174,** Lem. 8.1**]**) *If:*

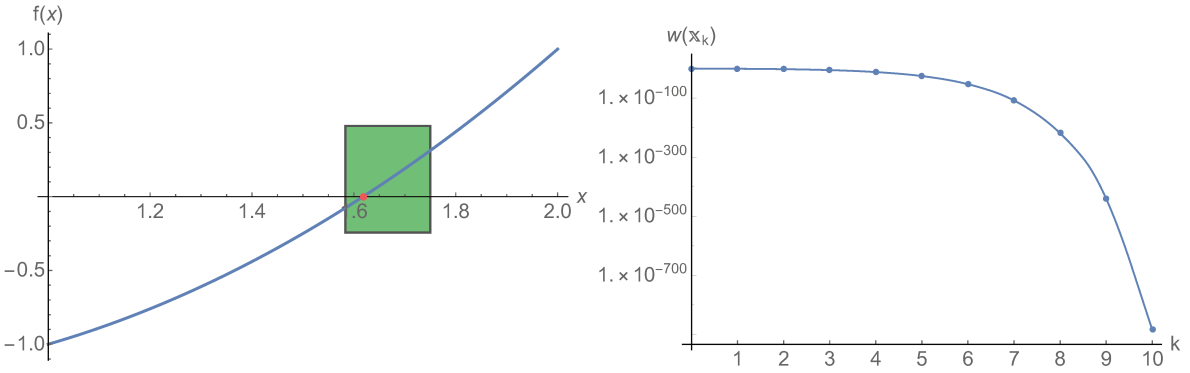
- *f is a rational function with a unique root and no pole in $\mathbb{i}$;*
- *$\mathbb{f}$ and $\mathbb{df}$ are the natural interval extensions of f and f';*
- *$\mathbb{f}(\mathbb{i}) \neq \perp\!\!\!\perp$, $\mathbb{df}(\mathbb{i}) \neq \perp\!\!\!\perp$ and $0 \notin \mathbb{df}(\mathbb{i})$;*

then there exists an initial interval $\mathbb{x}_0 \subseteq \mathbb{i}$ and a constant $C \geqslant 0$ such that

$$w(\mathbb{x}_{k+1}) \leqslant C w(\mathbb{x}_k)^2, \qquad k \geqslant 0.$$

The proof may be found in **[173,** Lem. 7.4**]**.

The exact computation with algebraic numbers described in **Section 1.1.3** emphasizes the importance of **Theorem 1.28**, for the latter provides as tight as desired rigorous interval enclosures required by the former. The example below illustrates the efficiency of Interval Newton's method for algebraic numbers.



(a) Plot of f over $[1,2]$; the enclosure of φ obtained at the first iteration is depicted in green.

(b) Width of interval $\mathbb{x}_k$ in function of k.

■ **Figure 1.7**: Interval Newton's method applied to $f : x \mapsto x^2 - x - 1$ with $\mathbb{x}_0 = [1,2]$, to isolate the golden ration φ. A typical quadratic convergence is observed.

■ **Example 1.29** (Golden ratio) *The* golden ratio $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$ *belongs to the most famous algebraic numbers, due to its appearance in various mathematical problems (Fibonacci numbers, construction of the regular pentagon, etc.), in architecture since antiquity, or even in some natural patterns like leaf arrangement*[7]*. It is the unique positive root of the polynomial*

$$f(x) = x^2 - x - 1.$$

The Interval Newton's method is quite simple and particularly efficient to compute highly accurate rigorous enclosures of φ with rational endpoints, when for example floating-point interval arithmetic is limited by the underlying floating-point precision.

Let $\mathbb{x}_0 = [1,2]$ be a first approximation of φ, and $\mathbb{f}, \mathbb{df}$ the natural $\mathbb{I}_{\mathbb{Q}}$-interval extensions of f, f'. For the first iteration, the method computes

$$\mathbb{f}([\mathrm{mid}(\mathbb{x}_0)]) = \mathbb{f}([\tfrac{3}{2}]) = [-\tfrac{1}{4}], \qquad \mathbb{df}(\mathbb{x}_0) = [1,3],$$
$$N(\mathbb{x}_0) = [\tfrac{3}{2}] \ominus [-\tfrac{1}{4}] \oslash [1,3] = [\tfrac{19}{12}, \tfrac{7}{4}], \qquad \mathbb{x}_1 = [\tfrac{19}{12}, \tfrac{7}{4}].$$

The rectangle $\mathbb{x}_1 \times \mathbb{f}(\mathbb{x}_1)$ is depicted on the plot in ***Figure 1.7a****. The decreasing of the interval widths $w(\mathbb{x}_k)$ is plotted in* ***Figure 1.7b****, exhibiting the typical curve of a quadratic rate of convergence – in accordance with* ***Theorem 1.28****. In fact, $\mathbb{x}_5$ already provides an accuracy of 10^{-25}, that is, for beyond standard double precision:*

$$\mathbb{x}_5 = \left[\frac{9174864818009113897476238962573654837791582221339033958 6}{5670378299715250742402011827769493404683560623404443812 5}, \right.$$
$$\left. \frac{3058288272672664992288436261997667727868832050508704037 4}{1890126099906913183773747676507325260650388480610030737 5}\right].$$

[7] Actually, its prestige at least partially results from abusive or esoteric considerations around it. Here we simply use: "φ is an algebraic number", which, after all, is its mere essence.

1.3.4 ▶ Limitations

In view of the foregoing, it might be tempting to use interval arithmetic everywhere, by replacing all floating-point numbers and operations by the interval analogues. Besides computational efficiency issues (clearly, interval arithmetic is at least twice slower than standard floating-point arithmetic for the same precision), a main problem is the sometimes important overapproximations of interval analysis, even in situations where the floating-point computations are quite reliable.

A major shortcoming of interval analysis is that the *correlations* between variables are lost, since in the end all quantities are overapproximated by their range. This limitation appears in the classical problematic cases given below. A more detailed survey of these shortcomings of interval arithmetic can be found in **[174, 247, 221]**. This emphasizes the need for more sophisticated *higher order methods*, which is addressed in the next chapters.

Range overapproximation. While natural interval extensions of functions have the advantage of providing an easy way to compute rigorous enclosures of the range, they may return extremely large and therefore useless approximations. Moreover, two different expressions for the same real-valued function lead to two different natural interval extensions, because $\mathbb{I}_{\mathbb{R}}$ does not satisfy all the relations in $\mathbb{R}$ (see **Proposition 1.20**), e.g. $x - x = 0$ holds for any $x \in \mathbb{R}$, but $[-a, a] \ominus [-a, a] = [-2a, 2a] \neq \mathbb{0}$ for $a > 0$.

One significant consequence is that the computed enclosure of the range of a function strongly depends on its *syntax tree*, that is, its concrete expression using mathematical symbols. Consider for example three different evaluation schemes for the same degree-2 polynomial (the standard form, the Horner form and the factorized form):

$$f_1(x) = x^2 - 2x + 1, \qquad f_2(x) = (x - 2)x + 1, \qquad f_3(x) = (x - 1)^2.$$

Let $\mathbb{i} = [-1, 2]$, we get using the corresponding natural interval extensions:

$$\mathbb{f}_1(\mathbb{i}) = [-3, 7], \qquad \mathbb{f}_2(\mathbb{i}) = [-5, 4], \qquad \mathbb{f}_3(\mathbb{i}) = [0, 4].$$

Only the last one gives the exact range. The second one gives the correct upper bound, but both $\mathbb{f}_1$ and $\mathbb{f}_2$ fail to prove the nonnegativity of f over $\mathbb{i}$.

Highly canceling functions are particularly subject to huge overapproximations, thereby forcing the user to subdivide the input interval to obtain a reasonable enclosure. Let us illustrate this phenomenon with the Taylor remainders for the cosine function,

$$f_n(x) = \left| \cos(x) - \sum_{i=0}^{2n} (-1)^i \frac{x^{2i}}{(2i)!} \right|, \qquad n \geqslant 0, \tag{1.3}$$

for which we compute a rigorous uniform upper bound m_n over $\mathbb{i} = [-1, 1]$ using the optimization method of **Section 1.3.2**, with a 10% tolerance. Bounding f_0 requires 20 subintervals, 192 are needed for f_1, 5,078 for f_2, 259,688 for f_3...! This highlights the limitations of such an approach to bound the error of accurate polynomial approximations.

Wrapping effect. The term *wrapping effect* **[184]** refers to all the overapproximation phenomena appearing when rigorously enclosing a region of $\mathbb{R}^n$ using intervals for the coordinates.

Indeed, Cartesian products of intervals only model hyperboxes of $\mathbb{R}^n$. This is particularly troublesome for problems involving iterations, like in dynamical systems theory.

We illustrate this phenomenon using a very elementary dynamical system in the plane, namely a rotation of some angle t:

$$f_t : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} \cos(t)x - \sin(t)y \\ \sin(t)x + \cos(t)y \end{pmatrix}, \qquad (x, y) \in \mathbb{R}^2.$$

Let $\mathbb{t}$ be a thin interval around t. Both components of f_t admit natural interval extensions using $\mathbb{cos}$ and $\mathbb{sin}$, yielding an interval extension $\mathbb{f}_{\mathbb{t}} : \mathbb{I}^2 \to \mathbb{I}^2$.

Let $B = [-1, 1] \times [-1, 1] \subseteq \mathbb{R}^2$. Then for all $k \geqslant 0$, $B_k := f_t^k(B)$ is a rotated square contained in the Euclidean centered ball of radius $\sqrt{2}$ (see **Figure 1.8a**). The interval version of this iteration is:

$$\mathbb{x}_0 = \mathbb{y}_0 := [-1, 1], \qquad \text{and} \qquad \begin{pmatrix} \mathbb{x}_{k+1} \\ \mathbb{y}_{k+1} \end{pmatrix} := \mathbb{f}_{\mathbb{t}} \begin{pmatrix} \mathbb{x}_k \\ \mathbb{y}_k \end{pmatrix}, \quad k \geqslant 0,$$

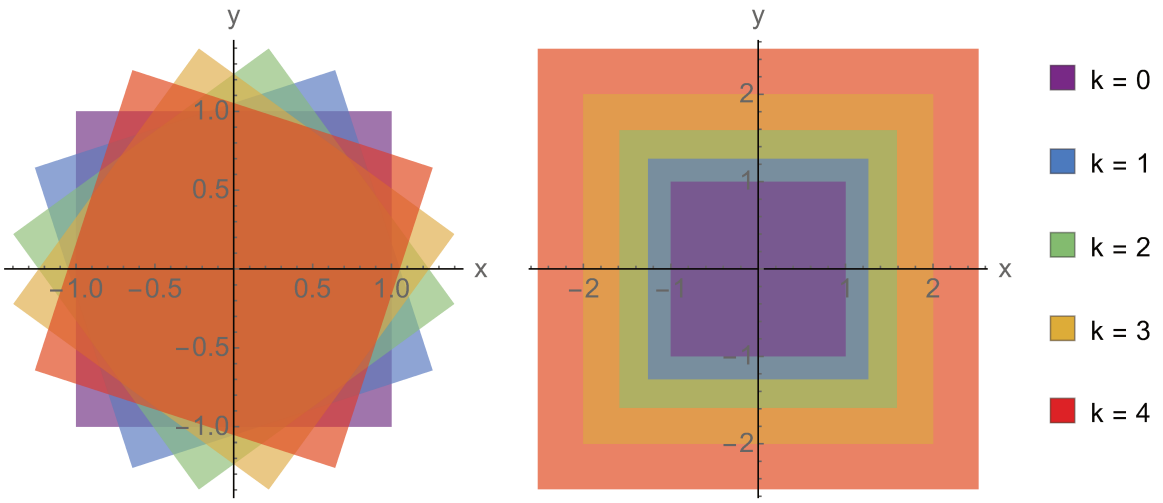
so that $B_k \subseteq \mathbb{x}_k \times \mathbb{y}_k$ for all k.

However, since a rotated square B_k can only be overapproximated by a box $\mathbb{x}_k \times \mathbb{y}_k$, each iteration increases the size of the box, as shown in **Figure 1.8a** for $t = \frac{\pi}{10}$. While the Euclidean norm of B_k always remains $\sqrt{2}$, **Figure 1.8b** shows the exponential grow of the magnitude of $\sqrt{\mathbb{x}_k^{②} \oplus \mathbb{y}_k^{②}}$. This is the *wrapping effect*.

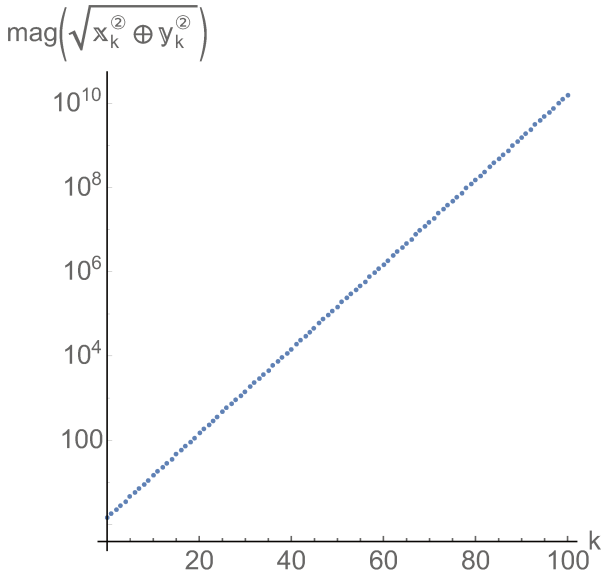
Matrix inversion. The inversion of matrices is a typical operation that requires a particular care when implemented with intervals. In most cases, replacing all floating-point operations by interval ones in a standard Gaussian elimination procedure is not a proficient solution. Indeed, the intervals tend to rapidly grow and the algorithm fails to find a pivot not containing zero, even if the initial matrix was far from singular. Of course, its is clear that ill-conditioned matrices induce large intervals, since those ones contain both the exact mathematical value and the floating-point evaluation. However, even well-conditioned matrices most of the time lead to the same problems, as illustrated by the example below.

In fact, rigorous inversion of matrices is never performed like this in practice, except for very small dimension. Instead, fixed-point based a posteriori methods are employed (see **Section 3.3.3**).

■ **Example 1.30** (Interval Gaussian elimination) *The Lehmer matrix* $L_n = \left(\frac{\min(i,j)}{\max(i,j)}\right)_{1 \leqslant i,j, \leqslant n}$ *is an example of a* well-conditioned *matrix, as illustrated by the error plot of* ***Figure 1.9a****. However, we observe that using Gaussian elimination produces large intervals and the procedure rapidly fails to find a pivot not containing* 0. ***Figure 1.9b*** *plots the minimal underlying floating-point precision needed for intervals to ensure that Gaussian elimination terminates, that is one pivot not containing zero was found at each iteration. Clearly, this method is inadequate for intensive rigorous computations with large matrices. More details on this pitfall can be found in* **[221,** Sec. 10.1**]**.

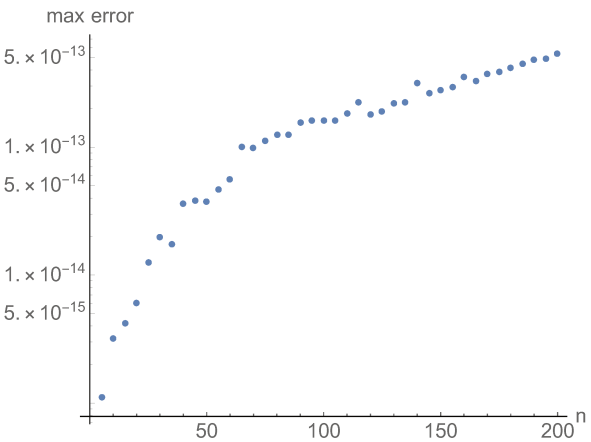


(a) Iterated rotations of a square: on the left, exact geometric transforms, on the right, rigorous enclosures of the x and y components with intervals.

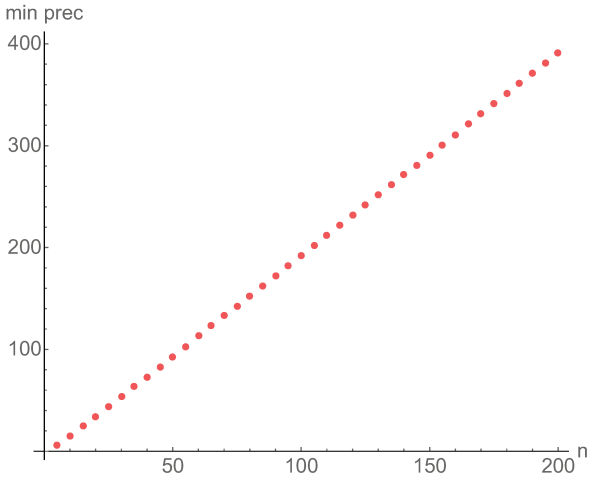


(b) The norm of $(\mathbb{x}_k, \mathbb{y}_k)$ grows exponentially with k.

■ **Figure 1.8**: Illustration of the wrapping effect during the rotation of a square in $\mathbb{R}^2$.



(a) Maximum absolute error in the entries of $L_n^{-1}L_n - \mathbf{1}_n$, evaluated using a Gaussian elimination procedure with machine double precision.



(b) Minimum floating-point precision needed so that the naive Gaussian elimination procedure with intervals does not fail.

■ **Figure 1.9**: Failure of naive Gaussian elimination with intervals on Lehmer matrices.

Computing with Functions

2

– Mais encore faut-il croire quelque chose dans le monde : qu'est-ce donc que vous croyez ?
– Ce que je crois ?
– Oui.
– Je crois que deux et deux sont quatre, Sganarelle, et que quatre et quatre sont huit.
— Molière, Don Juan ou le Festin de pierre

To a certain extent, computing with *functions* follows the same guidelines as computing with numbers, which was the subject of the preceding chapter. Indeed, elementary classes of functions admit exact representations together with algorithms to manipulate them: polynomials, rational functions, algebraic functions, etc. However, the algebraic theory of such functions is more complicated, and algorithms are often more costly than for numbers. **Section 2.1** is a short introduction to a particularly important – but maybe still not completely standard – class of functions: the *D-finite functions*, which will play a central role on many occasions throughout this thesis. These functions are defined as the solutions of linear ordinary differential equations (LODEs) with polynomial coefficients. Due to the efforts of an active community in this domain, efficient implementations available notably in Maple and Mathematica make it possible to manipulate them as *data structures*.

Yet, D-finite functions do not cover all the standard functions encountered in mathematics, a simple counter-example being the tan function. Hence, in order to efficiently compute with larger classes of functions, we resort to *approximate* representations of functions, in an analogy with floating-point numbers being used to approximate real numbers. However, the picture becomes more complex with functions: Which set of approximating functions do we choose? Polynomials are one possibility among others. How do we represent these polynomials, in which basis? Which norm do we use to quantify the quality of approximations? Some answers to these questions are provided in a condensed summary of approximation theory given in **Section 2.2**. This will be particularly helpful for **Chapters 4** and **5**, where Chebyshev expansions for D-finite functions are computed and validated. In contrast with this mathematical point of view about function approximation, **Chapter 8** gives an application where computer-specific aspects are taken into account, such as rounding errors due to the use of floating-point arithmetic.

By presenting both exact and approximate representations of functions, this chapter gives the necessary preliminaries before getting to the heart of the matter in **Chapter 3**: the *rigorous polynomial approximations*, that bring rigorous numerics to the level of *functions*, allowing for validated computations in function spaces, as well as intervals were used at the level of numbers.

2.1 D-finite functions: a successful example of exact representations

D-finite functions are the solutions of linear ordinary differential equations (LODEs) with polynomial coefficients (cf **Definition 2.1**). As mentioned in the introduction, a wide range of usual mathematical functions fall into this category: rational functions, algebraic functions, $\exp, \cos, \sin$ and their reciprocals, Bessel functions, etc. The fact that about 60% of functions listed in the famous *Handbook of mathematical functions* **[2]** are D-finite **[222]** is often put forward to advocate their central role in a wide range of applications.

D-finite functions (for *differentially finite functions*), were formally introduced in 1980 by Stanley **[234]**, although many of their properties had already been studied in the second half of the nineteenth century by Cauchy, Fuchs and Frobenius, among others. The development of modern and efficient computer algebra systems then increased the interest of mathematicians for them, and an active community rapidly emerged to embrace different aspects, such as connections with combinatorics, or summation/integration algorithms by Zeilberger's *Creative Telescoping* **[269, 59]**, which makes it possible to compute closed forms for expressions involving sums and integrals. This extended abstract **[223]** provides a broad overview of the fascinating properties and applications offered by D-finite functions.

The key idea about D-finite functions is that they can really be represented and manipulated as a *data structure* **[223]**, that is, a LODE with polynomial coefficients together with sufficiently many initial conditions. Based on that, the Maple package Gfun[1] **[224]** provides a user-oriented framework to efficiently compute with (univariate) D-finite functions. **Section 2.1.1** gives a brief overview of how this is possible, thanks to elementary definitions and properties about univariate D-finite functions. These notions will be useful in **Chapter 4**.

In a second time, **Section 2.1.2** addresses the more complicated theory of multivariate D-finite functions, which have also been implemented in Maple and Mathematica with the Mgfun[2] **[59]** and HolonomicFunctions[3] **[142]** packages, respectively. In particular, this is the starting point of a wide range of the aforementioned *Creative Telescoping* algorithms (**Section 2.1.3**). Multivariate D-finite functions and Creative Telescoping will be marginally used in **Chapter 6**, and they will "play the lead role" in **Chapter 9**.

2.1.1 ▶ D-finite functions: the univariate case

Throughout this section, $\mathbb{K}$ denotes a subfield of $\mathbb{C}$, usually a computable one, so that the manipulations and operations described below can be implemented, for instance, in computer algebra systems. This short introduction to D-finite functions, although sufficient for our purpose, can be supplemented by the reading of more thorough references, e.g., **[234]** or **[172,** Chap. 5**]**. Note that our presentation directly considers functions instead of formal power series, since the global framework of this thesis is mainly analytic, rather than algebraic.

[1] `http://perso.ens-lyon.fr/bruno.salvy/software/the-gfun-package/`
[2] `https://specfun.inria.fr/chyzak/mgfun.html`
[3] `https://www3.risc.jku.at/research/combinat/software/ergosum/RISC/HolonomicFunctions.html`

Definition 2.1 (D-finite function) *Let $f : U \to \mathbb{C}$ be an analytic function over a domain U of $\mathbb{C}$ (i.e., $U \subseteq \mathbb{C}$ is open and connected). Then f is said* differentially finite *(or, in short,* D-finite*) if there exist an order $r \geqslant 0$ and polynomials $p_0(x), \ldots, p_r(x) \in \mathbb{K}[x]$ with $p_r \neq 0$ such that:*

$$\mathbf{L} \cdot f(x) := p_r(x) f^{(r)}(x) + \cdots + p_1(x) f'(x) + p_0(x) f(x) = 0.$$

Equivalently, its derivatives $\{f^{(i)}, i \geqslant 0\}$ span a finite-dimensional space over $\mathbb{K}(x)$, the field of rational functions with coefficients in $\mathbb{K}$.

Remark 2.2 *The function f and its higher order derivatives are analytic, hence meromorphic, functions over the domain U. Therefore, seeing them as elements of a $\mathbb{K}(x)$-vector field makes sense.*

According to the Picard-Lindelöf theorem **[62,** Thm. I.3.1**]**, the equation $\mathbf{L} \cdot f = 0$ with prescribed initial conditions $f^{(i)}(x_0) = v_i$ for $i \in [\![0, r-1]\!]$ and some $x_0 \in U$, admits a unique solution. Therefore, a D-finite function is exactly represented by the data $(\mathbf{L}, U, x_0, v_0, \ldots, v_{r-1})$. This is the basis of the "*linear differential equations as data structure*" approach advocated in **[223]** and implemented in the Gfun package **[224]**.

The following theorem gives crucial closure properties of D-finite functions under usual operations, allowing for defining an arithmetic on those functions.

Theorem 2.3 *Let f, g be D-finite functions over a domain U of $\mathbb{C}$. Then:*

- *λf is D-finite, for any $\lambda \in \mathbb{K}$;*
- *$f + g$ is D-finite;*
- *fg is D-finite;*
- *f' is D-finite;*
- *any primitive F of f (i.e., $F' = f$) is D-finite;*
- *$f \circ h$ is D-finite for any algebraic function h over $\mathbb{K}$.*

Proofs of these properties may be found in **[234,** Thms. 2.3 and 2.7**]**. Roughly speaking, the key argument is that higher order derivatives of D-finite functions span a finite-dimensional linear subspace over $\mathbb{K}(x)$. For example, consider the addition of two D-finite functions f and g, of respective order r and s. First, $\{f^{(i)}, i \in [\![0, r-1]\!]\}$ and $\{g^{(j)}, j \in [\![0, s-1]\!]\}$ are generating families for the $\mathbb{K}(x)$-linear subspaces spanned by the derivatives of f and g, respectively. Now, the derivatives of $f + g$ are all contained in the subspace spanned by $\{f^{(i)} g^{(j)}, i, j \geqslant 0\}$, and the previous remark implies that $\{f^{(i)} g^{(j)}, i \in [\![0, r-1]\!], j \in [\![0, s-1]\!]\}$ is a generating family. Therefore, the derivatives of $f + g$ necessarily span a finite-dimensional subspace over $\mathbb{K}(x)$.

The proofs of these closure properties are translated into effective algorithms by the use of linear algebra to find $\mathbb{K}(x)$-linear relations between the derivatives of $f + g$, fg, etc., thus reconstructing a differential equation with polynomial coefficients.

Example 2.4 (Gfun in action) *The functions* exp *and* cos *are D-finite, since they respectively satisfy:*

$$\exp' - \exp = 0, \quad \exp(0) = 1,$$
$$\cos'' + \cos = 0, \quad \cos(0) = 1, \quad \cos'(0) = 0.$$

The Gfun *package for* Maple *represents these functions as a tuple containing the differential equation plus the associated initial conditions. Closure operations, implemented as algorithms, allows us to compute a differential equation for, e.g.,* $\exp(x^2)+\cos(x)$*:*

$$(4x^2+3)y'''+(-8x^3-14x)y''+(4x^2+3)y'+(-8x^3-14x)y=0,\quad y(0)=2,\ y'(0)=0,\ y''(0)=0.$$

Moreover, given an arbitrary univariate expression, the routine `holexprtodiffeq` *automatically tries to reconstruct such an equation by applying closure operations and recognizing known functions at the leaves of the expression.*

D-finite functions and P-recursive sequences

Another very important feature of D-finite functions is their connection with linear recurrences with polynomial coefficients, via their Taylor expansion. Specifically, let f be a D-finite function over a domain U, and $x_0 \in U$. By a simple translation, we assume that $x_0 = 0$. Since f is analytic over U, it has a unique power series expansion around 0:

$$f(x) = \sum_{n=0}^{+\infty} a_n x^n.$$

Injecting this *ansatz* in the differential equation $\mathbf{L}\cdot f = 0$ and formally differentiating the power series, yields a recurrence relation with polynomial coefficients in n on the a_n. Such a sequence $(a_n)_{n\geqslant 0}$ is called *P-recursive* [234].

Definition 2.5 (P-recursive sequence) *A sequence $(a_n)_{n\geqslant 0}$ of complex numbers is called* polynomially recursive *(or, in short,* P-recursive*) if there exist an order $r \geqslant 0$ and polynomials $c_0(n), \ldots, c_r(n) \in \mathbb{K}[n]$ with $c_r \neq 0$ such that:*

$$(\mathbf{R}\cdot a)_n = c_r(n)a_{n+r} + \cdots + c_1(n)a_{n+1} + c_0(n)a_n = 0, \qquad \text{for all } n \in \mathbb{N}.$$

The formal equivalence between D-finite functions (or their power series) and P-recursive sequences is established by the following theorem, stated and proved in [234, Thm. 1.5].

Theorem 2.6 *An analytic function f over a domain U containing 0 is D-finite if and only if its power series expansion at 0 is given by a P-recursive sequence $(a_n)_{n\geqslant 0}$.*

Example 2.7 *Consider the entire function $f(x) = e^{-x^2}\cos x$. Since* exp *and* cos *are clearly D-finite, so is f, using Theorem 2.3 for the product and the algebraic composition. The* Gfun *package for* Maple *automatically computes a differential equation, thanks to the routine* `holexprtodiffeq`*:*

$$f''(x) + 4xf'(x) + (4x^2+3)f(x) = 0.$$

Injecting a power series ansatz $f(x) = \sum_{n=0}^{+\infty} a_n x^n$ *in this equation gives:*

$$\sum_{n=0}^{+\infty} \left(n(n-1)a_n x^{n-2} + 4na_n x^n + 4a_n x^{n+2} + 3a_n x^n\right) = 0.$$

Identifying the powers of x in this sum yields the following recurrence relation for the P-recursive sequence (a_n):

$$(n+3)(n+4)a_{n+4} + (4n+11)a_{n+2} + 4a_n = 0, \qquad \text{for all } n \in \mathbb{N}.$$

Remark 2.8 (Expansions in other bases) *Though the most classical, the power series expansion of a D-finite function is not the only expansion for which the differential equation automatically translates into a recurrence relation for the coefficients. For example, the Chebyshev series expansion also has this property. Explicit recurrences are given in* **[194, 206, 18, 19]**, *whereas the linear system solved in* ***Chapter 4*** *implicitly encodes them.*

2.1.2 ▶ The multivariate case: D-finiteness vs holonomicity

When moving to the *multivariate* case, that is, functions $f(x_1, \ldots, x_n)$ depending on n complex variables $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{C}^n$, linear ODEs must be replaced by systems of PDEs (with polynomial coefficients). Contrary to the univariate case, two closely-related notions must be carefully distinguished: *D-finiteness* and *holonomicity*. Related technicalities go far beyond the scope of this manuscript. Detailed but still intuitive introductions are given in **[58, 141]**.

First, we define formal differential operators.

Definition 2.9 (Weyl algebra or polynomial Ore algebra) *The* n-th Weyl algebra $\mathfrak{D}_n := \mathbb{K}[\boldsymbol{x}]\langle\boldsymbol{\partial_x}\rangle = \mathbb{K}[x_1, \ldots, x_n]\langle\partial_{x_1}, \ldots, \partial_{x_n}\rangle$ *is the ring of differential operators with polynomial coefficients, generated by* $\{x_1, \ldots, x_n, \partial_{x_1}, \ldots, \partial_{x_n}\}$ *and quotiented by the relations:*

$$\partial_{x_i} x_j = \begin{cases} x_i \partial_{x_i} + 1, & i = j, \\ x_j \partial_{x_i}, & i \neq j, \end{cases} \qquad x_i x_j = x_j x_i, \quad \partial_{x_i} \partial_{x_j} = \partial_{x_j} \partial_{x_i}.$$

We have that $\{\boldsymbol{x}^{\boldsymbol{\beta}} \boldsymbol{\partial_x}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{N}^n\}$ is a basis of $\mathfrak{D}_n$ as a $\mathbb{K}$-vector space. If $L = \sum_{\boldsymbol{\alpha},\boldsymbol{\beta}} c_{\boldsymbol{\alpha},\boldsymbol{\beta}} \boldsymbol{x}^{\boldsymbol{\beta}} \boldsymbol{\partial_x}^{\boldsymbol{\alpha}}$, its *order* is the largest value of $|\boldsymbol{\alpha}|$ such that there exists $\boldsymbol{\beta}$ with $c_{\boldsymbol{\alpha},\boldsymbol{\beta}} \neq 0$.

Definition 2.10 (Rational Ore algebra) *The* rational Ore algebra $\mathfrak{D}_n^* := \mathbb{K}(\boldsymbol{x})\langle\boldsymbol{\partial_x}\rangle = \mathbb{K}(x_1, \ldots, x_n)\langle\partial_1, \ldots, \partial_n\rangle$ *is the ring of differential operators with rational fraction coefficients, where the commutation rules of* $\mathfrak{D}_n$ *are extended by*

$$\partial_{x_i} q(\boldsymbol{x}) = q(\boldsymbol{x}) \partial_{x_i} + \frac{\partial q(\boldsymbol{x})}{\partial x_i}, \qquad q(\boldsymbol{x}) \in \mathbb{K}(\boldsymbol{x}).$$

Differential operators in $\mathfrak{D}_n$ naturally act on smooth functions via $\partial_{x_i} \cdot f = f_{x_i} := \frac{\partial f}{\partial x_i}$. The *annihilator* $\mathfrak{Ann}(f)$ is a left ideal of $\mathfrak{D}_n$:

$$\mathfrak{Ann}(f) := \{L \in \mathfrak{D}_n \mid L \cdot f = 0\}.$$

One can also see $\mathfrak{Ann}(f)$ as a left ideal of $\mathfrak{D}_n^*$, and the quotient $\mathfrak{D}_n^*/\mathfrak{Ann}(f)$ as a $\mathbb{K}(\boldsymbol{x})$-vector space. Roughly speaking, a function f is D-finite if $\mathfrak{Ann}(f)$ contains enough equations so that f can be uniquely characterized by a finite set of initial conditions.

Definition 2.11 *An analytic function* f *over a domain* U *of* $\mathbb{C}^n$ *is called D-finite if* $\mathfrak{D}_n^*/\mathfrak{Ann}(f)$ *has finite dimension. Equivalently, its higher order derivatives* $\{\boldsymbol{\partial_x}^{\boldsymbol{\alpha}} \cdot f, \boldsymbol{\alpha} \in \mathbb{N}^n\}$ *form a finite-dimensional vector space over the field* $\mathbb{K}(\boldsymbol{x})$ *of rational fractions.*

In other cases, when f is a "generalized function", for instance a distribution, $\mathfrak{Ann}(f)$ can only be seen as a left ideal of $\mathfrak{D}_n$ and $\mathfrak{D}_n/\mathfrak{Ann}(f)$ as a $\mathbb{K}$-vector space. For example, the univariate Dirac distribution, defined by $\langle \delta, f\rangle = f(0)$, is annihilated (as a distribution) by x, since $\langle x\delta, f\rangle = \langle \delta, xf\rangle = 0$. However, a left ideal of $\mathfrak{D}_1^*$ containing x is necessarily $\mathfrak{D}_1^*$, but 1 annihilating δ would imply $\delta = 0$. In that setting, the relevant notion is *holonomicity*, arising in Bernstein's D-module theory [21, 85].

Definition 2.12 *Let $\mathfrak{I}$ be a left ideal of $\mathfrak{D}_n$. For $L \in \mathfrak{D}_n$, let $[L]_\mathfrak{I}$ denote the class of L in the quotient $\mathfrak{D}_n/\mathfrak{I}$. For $s \geqslant 0$, define*

$$\left(\mathfrak{D}_n/\mathfrak{I}\right)_s = \mathrm{Span}_\mathbb{K}\left\{[\boldsymbol{x}^{\boldsymbol{\beta}}\boldsymbol{\partial}_{\boldsymbol{x}}^{\boldsymbol{\alpha}}]_\mathfrak{I}, |\boldsymbol{\alpha}| + |\boldsymbol{\beta}| \leqslant s\right\}.$$

Then there exists a polynomial $b(s) \in \mathbb{K}[s]$ such that $\dim_\mathbb{K}(\mathfrak{D}_n/\mathfrak{I})_s = b(s)$ for s large enough. The degree of $b(s)$ is called the Bernstein dimension *of $\mathfrak{D}_n/\mathfrak{I}$. The left ideal $\mathfrak{I}$ is called holonomic if the Bernstein dimension of $\mathfrak{D}_n/\mathfrak{I}$ is equal to n.*

An analytic function f over a domain U of $\mathbb{C}^n$ is said to be holonomic if the $\mathfrak{D}_n$-left ideal $\mathfrak{Ann}(f)$ is holonomic.

The algorithmic treatment of multivariate D-finite functions is more complex than the univariate case. It requires the use of *non-commutative Gröbner bases*, which are a generalization of the classical Gröbner bases for polynomial systems (see for example [85, 59, 140] and references therein). Roughly speaking, the idea consists in fixing an appropriate total order on the monomials $\{\boldsymbol{x}^{\boldsymbol{\beta}}\boldsymbol{\partial}_{\boldsymbol{x}}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{N}^n\}$ (in the polynomial case) or $\{\boldsymbol{\partial}_{\boldsymbol{x}}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha} \in \mathbb{N}^n\}$ (in the rational case). Then, given an input set of generators for an $\mathfrak{D}_n$ (or $\mathfrak{D}_n^*$)-left ideal $\mathfrak{I}$, the *Buchberger algorithm* computes a new generating set of the basis with the property that the reduction of any differential operator L modulo $\mathfrak{I}$ using this basis and the monomial order is convergent. This can be seen as an extension of the Euclidean division to this multivariate non-commutative setting.

Similarly to the univariate case, closure properties for D-finite [59, 140] and holonomic functions [85, 240, 188] under usual operations are available. Using variants of the *FGLM algorithm* [140], which transforms a Gröbner basis for a given monomial ordering into a new one for another specified ordering, corresponding D-finite or holonomic annihilating ideals can moreover be automatically computed. Such manipulations are implemented for example in the Maple package Mgfun [59] and in the Mathematica package HolonomicFunctions [142].

In Chapter 9, we will also need multi-indexed sequences satisfying linear recurrences with polynomial coefficients. Such recurrence (or difference) operators are defined as follows.

Definition 2.13 *$\mathfrak{R}_n := \mathbb{K}[\boldsymbol{\alpha}]\langle \boldsymbol{S}_{\boldsymbol{\alpha}}\rangle = \mathbb{K}[\alpha_1, \ldots, \alpha_n]\langle S_{\alpha_1}, \ldots, S_{\alpha_n}\rangle$ is the set of difference operators with polynomial coefficients in $\boldsymbol{\alpha}$, acting on sequences $u = (u(\gamma_1, \ldots, \gamma_n))_{\gamma \in \mathbb{N}^n}$ via*

$$\begin{aligned}(\alpha_i \cdot u)(\gamma_1, \ldots, \gamma_n) &= \gamma_i u(\gamma_1, \ldots, \gamma_n),\\ (S_{\alpha_i} \cdot u)(\gamma_1, \ldots, \gamma_n) &= u(\gamma_1, \ldots, \gamma_i + 1, \ldots, \gamma_n), \qquad \boldsymbol{\gamma} \in \mathbb{N}^n.\end{aligned}$$

The annihilator $\mathfrak{Ann}(u) = \{R \in \mathfrak{R}_n \mid R \cdot u = 0\}$ is the set of recurrence relations satisfied by u, which is holonomic when its generating series is holonomic [59].

Note that continuous and discrete variables can be used together, yielding functions $f(\boldsymbol{x}, \boldsymbol{\alpha}) = f(x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m)$ with $x_i \in \mathbb{C}$ and $\alpha_j \in \mathbb{N}$. In such a case, the associated Ore algebra:

$$\mathbb{K}(\boldsymbol{x}, \boldsymbol{\alpha})\langle \boldsymbol{\partial}_{\boldsymbol{x}}, \boldsymbol{S}_{\boldsymbol{\alpha}}\rangle = \mathbb{K}(x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m)\langle \partial_{x_1}, \ldots, \partial_{x_n}, S_{\alpha_1}, \ldots, S_{\alpha_m}\rangle,$$

contains *differential-difference operators*, defined from ∂_{x_i} and S_{α_j}, as in the toy example below.

Example 2.14 (Moments of the Gaussian distribution) *Consider the function*

$$f(x,n) := x^n \frac{e^{-x^2/2}}{\sqrt{2\pi}}, \qquad \text{for } x \in \mathbb{C} \text{ and } n \in \mathbb{N},$$

which is the integrand of the formula defining the moments of the centered and normalized Gaussian distribution, as it will be seen in ***Example 2.15****.*

Its annihilator $\mathfrak{Ann}(f)$ *is generated by these two differential-difference operators:*

$$S_n - x, \qquad x\partial_x + (x^2 - n).$$

Since $\mathbb{K}(x,n)\langle \partial_x, S_n\rangle/\mathfrak{Ann}(f)$ *is of dimension* 1 *(generated by the class of* 1*),* f *is D-finite.*

2.1.3 ▶ Creative Telescoping

An interesting feature of the multivariate case is to integrate with respect to a continuous variable, or to sum with respect to a discrete variable. For a function $f(x_1,\ldots,x_n,\alpha_1,\ldots,\alpha_m)$, one can define, provided that integration and summation operations are well-defined:

$$\varphi_i(\widehat{\boldsymbol{x}}_i,\boldsymbol{\alpha}) := \int_{a_i}^{b_i} f(\boldsymbol{x},\boldsymbol{\alpha})\mathrm{d}x_i, \qquad i \in [\![1,n]\!], \qquad a_i,b_i \in \mathbb{R}\cup\{\pm\infty\}, \tag{2.1}$$

$$\psi_j(\boldsymbol{x},\widehat{\boldsymbol{\alpha}}_j) := \sum_{\alpha_j=\beta_j}^{\gamma_j} f(\boldsymbol{x},\boldsymbol{\alpha}), \qquad j \in [\![1,m]\!], \qquad \beta_j,\gamma_j \in \mathbb{N}\cup\{+\infty\}.$$

where $\widehat{\boldsymbol{x}}_i := (x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n) \in \mathbb{C}^{n-1}$ and $\widehat{\boldsymbol{\alpha}}_j := (\alpha_1,\ldots,\alpha_{j-1},\alpha_{j+1},\ldots,\alpha_m) \in \mathbb{N}^{m-1}$.

Originating from the seminal work of Zeilberger [269], several algorithms identified under the name *Creative Telescoping* were developed to compute D-finite and/or holonomic annihilators for such functions φ_i,ψ_j, provided the input function f is given by such an annihilator [240, 58, 141]. A historical survey on that topic may be found in [59].

We briefly sketch out the key idea. To construct elements of $\mathfrak{Ann}(\varphi_i)$ with φ_i defined in Equation (2.1), we look for operators $T \in \mathfrak{Ann}(f)$ of the form:

$$T := P + \partial_{x_i} Q = 0 \qquad \bmod \mathfrak{Ann}(f), \tag{2.2}$$

with the additional condition that P does not depend on x_i nor ∂_{x_i}, so that it commutes with the integration operation $\int_{a_i}^{b_i} \ldots \mathrm{d}x_i$, yielding the following equation for φ_i:

$$P\cdot\varphi_i(\widehat{\boldsymbol{x}}_i,\boldsymbol{\alpha}) + [f(\boldsymbol{x},\boldsymbol{\alpha})]_{x_i=a_i}^{x_i=b_i} = 0,$$

where the bracket expression above is to be considered with limits if a_i or b_i is $\pm\infty$. In many situations, where a_i and b_i are called *natural boundaries* [240], this bracket vanishes, so that $P \in \mathfrak{Ann}(\varphi_i)$. Otherwise, one must find another operator R in the annihilator of this bracket, so that $RP \in \mathfrak{Ann}(\varphi_i)$.

Example 2.15 (Moments of the Gaussian distribution – Creative Telescoping) *Consider again the function f of **Example 2.14**. By integrating it for x from $-\infty$ to $+\infty$, we obtain the moments m_n of the Gaussian distribution, with the following closed form expression* [197, Sec. 5.4]*:*

$$m_n := \int_{-\infty}^{+\infty} f(x,n)\mathrm{d}x = \begin{cases} \frac{(2k)!}{2^k k!} & \text{if } n = 2k, \\ 0 & \text{if } n = 2k+1. \end{cases} \tag{2.3}$$

*Creative Telescoping offers an elegant way to check or recover this expression. By permuting x and ∂_x in the second generator of $\mathfrak{Ann}(f)$ found in **Example 2.14**, we have $\partial_x x + (S_n^2 - (n+1)) \in \mathfrak{Ann}(f)$. Using the first generator, we obtain an operator of the prescribed form (2.2):*

$$\big(S_n^2 - (n+1)\big) + \partial_x x \quad \in \quad \mathfrak{Ann}(f).$$

Integrating this relation for x over $(-\infty, +\infty)$ and noticing that $xf(x,n) \to 0$ as $x \to \pm\infty$ for any n, we obtain the following recurrence of order 2 for m_n:

$$m_{n+2} - (n+1)m_n = 0, \qquad \text{for all } n \in \mathbb{N}.$$

This recurrence relation, together with $m_0 = 1$ and $m_1 = 0$, has the closed form of Equation (2.3) as unique solution.

2.2 A condensed summary of approximation theory

Broadly speaking, approximation theory is the art of providing an accurate and reliable representation of a given function using (combinations of) simpler functions, called approximating functions. The input function may be given as a symbolic expression, a table of sample values or moments, a solution of a differential equation or other types of functional equations. Approximating functions usually have strong properties and are convenient to compute with. Polynomials are probably the simplest class of such approximating functions, and also one of the most widely used. This thesis mainly focuses on them. Trigonometric functions also play an important role, especially when the input function has a natural periodic behavior. Exponential functions, Bessel functions, and others also appear in some specific domains like signal processing, theoretical physics or probability theory.

What are the advantages of such approximate representations over the input data itself? First, if the input function is supposed to be regular enough but only given through a limited quantity of information (a few sample values, for example), the approximation constructed out of it may faithfully represent the function *everywhere*. Then, even in the case where a lot of sample values are known, approximations often have the advantage of being a smooth, compact and easy-to-compute-with representation. In some sense, approximation theory makes it possible to actually compute with functions, in analogy with floating-point arithmetic for real number computations.

A natural question is: How do we define a *good* approximation? After a presentation of the general setting and a short reminder of Taylor approximations in **Section 2.2.1**, this question is addressed in two different frameworks: first, uniform approximation and the minimax theory in **Section 2.2.2**; second, least-squares (or L^2) approximation in **Section 2.2.3**, leading to orthogonal systems of polynomials. At this point, we introduce the well-known Chebyshev polynomials, for they will repeatedly play a central role throughout the thesis. More specifically, **Section 2.2.4** is concerned with Chebyshev series expansion, while **Section 2.2.5** deals with Chebyshev interpolation, very useful in practical implementations.

2.2.1 ▶ General setting

We fix a normed linear space $(\mathcal{F}, \|\cdot\|)$ of real-valued *input functions*, defined over the (open or closed) real interval I of left and right extremities $a, b \in \mathbb{R} \cup \{-\infty, +\infty\}$. Functions in $\mathcal{F}$ are to be approximated by elements of the linear subspace $\mathcal{A} \subseteq \mathcal{F}$, called the set of *approximating functions*. This sections tries to be as general as possible concerning the set $\mathcal{A}$ of approximating functions, even if polynomial approximations will be mainly considered throughout this manuscript, as indicated by the title.

In most settings, $\mathcal{A}$ is an infinite-dimensional subspace, defined as the monotone union $\mathcal{A} = \bigcup_{n \geqslant 0} \mathcal{A}_n$ of finite-dimensional subspaces $\mathcal{A}_n$, where n may be seen as a precision parameter. A typical example is the set of real-valued polynomials graduated by the degree:

$$\mathcal{A} = \mathbb{R}[x] = \bigcup_{n \geqslant 0} \mathcal{A}_n \qquad \text{where } \mathcal{A}_n = \mathbb{R}_n[x] = \{p \in \mathbb{R}[x] \mid \deg p \leqslant n\}.$$

This raises the following crucial questions:

- If $p_n^* \in \mathcal{A}_n$ denotes the best approximation of a given $f \in \mathcal{F}$ in $\mathcal{A}_n$ (if any), do we have $p_n^* \to f$ as $n \to +\infty$ in $\mathcal{F}$, that is $\|p_n^* - f\| \to 0$?
- If yes, what is the asymptotic rate of convergence?

Those questions, among others, will be treated in the following for two notions of convergence, induced by two different norms:

- **Section 2.2.2** is devoted to *uniform approximation*, where the objective is to minimize the maximal approximation error encountered over the interval of interest I.
- **Section 2.2.3** focuses on L^2 *approximation*, where the quantity to minimize is the quadratic error defined as the integral of the squared error function.

Before addressing these two topics, we provide a brief reminder on a very classical and natural type of approximations, namely Taylor expansions. Despite their quite elegant properties, their intrinsic limitations motivate the need of better approximation tools.

A brief reminder on Taylor expansions

Let $f : I \to \mathbb{R}$ be a real-valued function defined over an interval I, such that f is n times differentiable at a given point $x_0 \in I$. Its Taylor series of order n at x_0 is defined as:

$$\mathbf{T}_n \cdot f(x) := \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k.$$

Roughly speaking, $\mathbf{T}_n \cdot f$ is the best order n approximation of f over an infinitely small neighborhood of x_0. The following theorem, called the Taylor-Lagrange estimation of the remainder, is a central property of Taylor approximations:

Theorem 2.16 (Taylor-Lagrange) *If f is a $n+1$ times differentiable real-valued function over an interval I containing x_0, then for all $x \in I \setminus \{x_0\}$, there exists $\xi \in (x_0, x)$ or (x, x_0) such that:*

$$f(x) - \mathbf{T}_n \cdot f(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1}.$$

In particular, if f is of class $\mathcal{C}^{n+1}$ over a compact segment $I = [a, b]$ with $\delta(I, x_0) = \max(|a - x_0|, |b - x_0|)$:

$$\|f - \mathbf{T}_n \cdot f\|_{\infty,I} \leqslant \frac{\|f^{(n+1)}\|_{\infty,I}}{(n+1)!}\delta(I, x_0)^{n+1}.$$

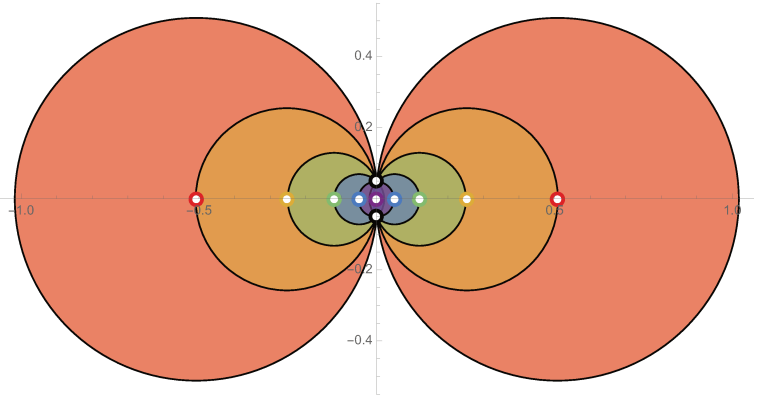
When f can be extended to a holomorphic function over the complex plane, an explicit geometric convergence rate of the Taylor series can be given:

Theorem 2.17 (Cauchy estimate [217, Thm. 10.15]) *Let f be a holomorphic function over an open disc of center z_0 and radius $R > 0$ in the complex place, denoted by $B(z_0, R)$, and bounded by $M \geqslant 0$ over $\partial B(z_0, r) = \{z \in \mathbb{C}, |z| = R\}$. Then:*

$$\left|\frac{f^{(n)}(z)}{n!}\right| \leqslant \frac{M}{R^n}, \qquad \text{for all } z \in B(z_0, R), n \geqslant 0, \text{ and}$$
$$|f(z) - \mathbf{T}_n \cdot f(z)| \leqslant \frac{M|z - z_0|}{R - |z - z_0|}\left(\frac{|z - z_0|}{R}\right)^n.$$

This theorem also puts into evidence the intrinsic limitation of Taylor approximations: the radius of convergence is given by the nearest singularity in the complex plane, even if our initial problem only considered the function f over a segment of the real line.

Example 2.18 *Consider the function $f_a : x \mapsto \frac{1}{a^2+x^2}$, for some $a > 0$. Although f_a is analytic over the whole real line, it has singularities in the complex plane: $\mathrm{i}a$ and $-\mathrm{i}a$. Since power series expansions are limited by the closest complex singularity, approximating f_a over, for instance, $[-1, 1]$, requires subdividing the interval and computing several Taylor expansions (see **Figure 2.1**).*



■ **Figure 2.1**: Convergence disks for Taylor expansions of $f_a(x) = 1/(a^2 + x^2)$ over $[-1, 1]$, with $a = 0.05$.

2.2.2 ▶ Uniform approximation: the minimax theory

In the context of uniform approximation, we consider $\mathcal{F} := \mathcal{C}^0([a, b])$, the space of real-valued continuous functions defined on the compact interval $I = [a, b]$. We endow $\mathcal{C}^0([a, b])$ with the norm $\|\cdot\| := \|\cdot\|_{\infty,[a,b]}$, called the *uniform norm* (or infinity, or supremum, or Chebyshev norm):

$$\|f\|_{\infty,[a,b]} = \max_{x\in[a,b]} |f(x)|,$$

which is finite and reached by at least one point $x \in [a, b]$, by compactness. This makes $(\mathcal{C}^0([a, b]), \|\cdot\|_{\infty,[a,b]})$ a Banach space, that is, a complete normed linear space **[53,** Sec. 1.3**]**.

Uniform approximation aims at minimizing the *maximum approximation error* $\|f - p\|_{\infty,[a,b]}$ between the input function f and an approximation p, assuming for example that $p \in \mathcal{A}_n$ for some given $n \geqslant 0$. The first positive result to be mentioned is the Weierstrass approximation theorem **[53,** Sec. 3.3**]**:

■ **Theorem 2.19** (Weierstrass) *Let $a \leqslant b \in \mathbb{R}$. The set of polynomials $\mathbb{R}[x]$ is dense in $(\mathcal{C}^0([a, b]), \|\cdot\|_{\infty,[a,b]})$.*

This means that if f is continuous over $[a, b]$, then there exists a sequence of polynomials $p_n \in \mathbb{R}_n[x]$ such that $\|f - p_n\|_{\infty,[a,b]} \to 0$ as $n \to +\infty$.

This crucial theorem admits several extensions, including the Stone-Weierstrass and Bishop approximation theorems for other sets $\mathcal{A}$ of approximating functions (see **[53,** Sec. 6.1**]** and **[218,** Sec. 5.7**]**).

Since the subspaces $\mathcal{A}_n$ are finite-dimensional, clearly any function $f \in \mathcal{F}$ admits a best uniform approximation $p_n^* \in \mathcal{A}_n$. However, this does not suffice to assert its uniqueness. For that, we need to introduce the notions of *Haar condition* and *Chebyshev system* **[53,** Sec. 3.4**]**.

■ **Definition 2.20** (Haar condition **[53,** Sec. 3.4**]**) *A set $A = \{\varphi_0, \varphi_1, \ldots, \varphi_m\}$ of $m + 1$ continuous functions over I satisfies the* Haar condition *if and only if any nontrivial linear combination of them cannot vanish more than m times on I. Such a system is called a* Chebyshev system.

Example 2.21 *The set* $\{1, x, \ldots, x^m\}$ *or any set of* $m+1$ *polynomials spanning* $\mathbb{R}_m[x]$ *is a Chebyshev system. Other examples are the trigonometric polynomials* $\{1, \cos t, \sin t, \ldots, \cos nt, \sin nt\}$ *over* $[0, 2\pi]$, $\{e^{\alpha_0 x}, \ldots, e^{\alpha_m x}\}$ *on* $\mathbb{R}$ *for* $\alpha_0 < \cdots < \alpha_m$.

Theorem 2.22 (Alternation criterion **[53,** Secs. 3.4 and 3.5**]**) *Let* $f \in \mathcal{C}^0([a,b])$ *and* $n \geqslant 0$. *We assume that* $\mathcal{A}_n$ *is a Chebyshev system. Then* f *admits a unique best approximation* $p_n^* \in \mathcal{A}_n$, *which is the unique element* $p \in \mathcal{A}_n$ *such that the approximation error* $f - p$ *satisfies the* equioscillation property*: there exist* $n+2$ *points* $a \leqslant x_0 < \cdots < x_{n+1} \leqslant b$ *such that:*

$$f(x_k) - p(x_k) = (-1)^k (f(x_0) - p(x_0)) = \pm \|f - p\|_{\infty,[a,b]}, \qquad k \in [\![0, n+1]\!].$$

p_n^* *is called the* minimax approximation *of* f *in* $\mathcal{A}_n$.

It is obvious that the use of floating-point arithmetic will not make it possible to achieve the exact statement of **Theorem 2.22**. More generally and beyond numerical errors, the approximation p to be certified may be only a *near-best* approximation (see below), and the next theorem is useful to bound the approximation defect.

Theorem 2.23 (La Vallée Poussin **[53,** Sec. 3.4**]**) *Let* $f \in \mathcal{C}^0([-1,1])$ *and* $p \in \mathcal{A}_n$, *assuming that* $\mathcal{A}_n$ *is a Chebyshev system. If there exist* $n+2$ *points* $-1 \leqslant x_0 < \cdots < x_{n+1} \leqslant 1$ *such that the approximation error* $f - p$ *alternates sign at the* x_i, *that is:*

$$(f(x_k) - p(x_k))(f(x_{k+1}) - p(x_{k+1})) \leqslant 0, \qquad k \in [\![0, n]\!],$$

then the following enclosure of the optimal approximation error holds, where p_n^* *is the minimax approximation of* f *in* $\mathcal{A}_n$*:*

$$\min_{0 \leqslant k \leqslant n+1} |f(x_k) - p(x_k)| \leqslant \|f - p_n^*\|_{\infty,[a,b]} \leqslant \|f - p\|_{\infty,[a,b]}.$$

Remez algorithm

Although **Theorems 2.22** and **2.23** provide explicit conditions to check the (near) optimality of candidate approximations p, they do not explain how to construct such approximations. An exchange algorithm published by Remez **[209, 208]** in 1934 allows for constructing degree n approximations of f, as close as desired to the optimal one, by means similar to polynomial interpolation (see **Section 2.2.5**). We only give the pseudo-code of this method (**Algorithm** Remez), and we refer the reader to **[202,** Chaps. 8 and 9**]** or **[53,** Sec. 5.8**]** for more details, in particular for its quadratic rate of convergence under some mild assumptions on f (the number of bits of precision doubles at each iteration). This algorithm has been extensively used in signal processing applications **[193, 82]**, or, closer to our interest, for elementary function implementation **[43, 54, 176]**.

However, its theoretical efficiency must be balanced by its behavior in practice with floating-point arithmetic. In particular, the choice of initial points has a major impact on the actual convergence of the algorithm with floating-point numbers **[82,** Sec. 3.5**]**. Moreover, for elementary function implementation using polynomials, the output of Remez algorithm is not guaranteed to be a good approximation, once the coefficients have been truncated to the target floating-point precision, nor to be accurately evaluated using the Horner scheme, due to the rounding errors. The first problem was addressed, e.g., in **[43]**, using integer programming and lattice reduction. The latter is the object of **Chapter 8**, which proposes a generalization of **Algorithm** Remez taking into account the (linearized) rounding error during evaluation.

Algorithm 2.1 REMEZ(f, n, Δ) – Remez second algorithm

Input: a function $f \in \mathcal{C}^0([-1,1])$, a natural integer n, and a tolerance Δ.
Output: An approximation p of the degree n-minimax polynomial of f.

1: Choose $n+2$ points $-1 \leqslant x_0 < x_1 < \cdots < x_{n+1} \leqslant 1$, $\delta \leftarrow 1, \varepsilon \leftarrow 0$.
2: **while** $\delta \geqslant \Delta|\varepsilon|$ **do**
3: Determine the solutions $a_0, \ldots, a_n$ and ε of the linear system:

$$\sum_{k=0}^{n} a_k x_j^k - f(x_j) = (-1)^j \varepsilon, \quad j \in [\![0, n+1]\!].$$

4: Choose $x_{\text{new}} \in [-1,1]$ such that:

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

5: Replace one of the x_i with x_{new}, in such a way that the sign of $p - f$ alternates at the points of the resulting discretization $x_0, \ldots, x_{n+1}$.
6: $\delta \leftarrow |p(x_{\text{new}}) - f(x_{\text{new}})| - |\varepsilon|$.
7: **end while**
8: Return p.

Near-optimal approximations

In a wide range of applications, approximations are not constructed to be optimal, but only *near-optimal*. Relaxing this optimality condition allows for more eficient methods than the Remez algorithm for instance. An important class of such methods are the *linear projections*, that is, bounded linear operators $\mathbf{L}_n : \mathcal{C}^0(I) \to \mathcal{A}_n$ satisfying $\mathbf{L}_n \circ \mathbf{L}_n = \mathbf{L}_n$ for all $n \geqslant 0$ ($\mathbf{L}_n$ preserves all elements in $\mathcal{A}_n$). Orthogonal truncated series (**Sections 2.2.3** and **2.2.4**) and polynomial interpolation (**Section 2.2.5**) are typical examples.

If $\mathbf{L}_n$ is a linear projection, then we call *Lebesgue constant* its operator norm associated to the $\|\cdot\|_\infty$ norm:

$$\Lambda_n := \|\mathbf{L}_n\|_\infty = \sup_{\substack{f \in \mathcal{C}^0(I), \\ f \neq 0}} \frac{\|\mathbf{L}_n \cdot f\|_{\infty,I}}{\|f\|_{\infty,I}} < +\infty$$

The Lebesgue constant is used to bound the overapproximation of $p_n = \mathbf{L}_n \cdot f$ to f compared to the minimax approximation p_n^* (see **[243,** Chap. 15**]** and **[53,** p. 147**]**). Indeed, since $\mathbf{L}_n \cdot (f - p_n^*) = p_n - p_n^*$, we have by the triangle inequality:

$$\|f - p_n\|_{\infty,I} \leqslant (1 + \Lambda_n)\|f - p_n^*\|_{\infty,I}$$

Moreover, combining this result with the general fact $\|f - p_n^*\|_\infty = O(1/n)$ for any Lipschitz function f over $[-1,1]$ (see **[53,** Jackson's Thm V (p. 147)**]**), we deduce that $\|f - p_n\|_\infty = O((1+\Lambda_n)/n)$. Hence, the asymptotic of Λ_n determines whether $p_n = \mathbf{L}_n \cdot f$ uniformly converges to f.

2.2.3 ▶ L^2 approximation and generalized Fourier series

In the context of L^2 approximation, one is interested in minimizing the *quadratic error*. We consider an interval I of $\mathbb{R}$ and a function $w : I \to \mathbb{R}$ positive almost everywhere. The 2-norm of a measurable function f defined on I, associated to the weight function w, is:

$$\|f\|_{2,w} = \left(\int_I f(x)^2 w(x) \mathrm{d}x \right)^{\frac{1}{2}}.$$

We define $\mathcal{F} = L^2_w(I)$, the space of square integrable functions f against $w(x)\mathrm{d}x$, that is $\|f\|_{2,w} < +\infty$. $L^2_w(I)$ is complete with respect to $\|\cdot\|_{2,w}$.

Then, for $f \in L^2_w(I)$, our objective is to find $p \in \mathcal{A}_n$ minimizing the quadratic error $\|f-p\|_{2,w}$. Contrary to uniform approximation which focuses on the maximal error, the $\|\cdot\|_{2,w}$ norm measures in some way a "cumulated error" or an "energy level". Applications are numerous, e.g., in signal processing.

Contrary to the $\|\cdot\|_{\infty,I}$, the $\|\cdot\|_{2,w}$ is associated to an inner product:

$$\langle f, g \rangle_w = \int_I f(x) g(x) w(x) \mathrm{d}x,$$

which makes $(L^2_w(I), \langle \cdot, \cdot \rangle_w)$ a Hilbert space.

Definition 2.24 *A countable family* $\{\varphi_0, \varphi_1, \ldots, \varphi_m, \ldots\}$ *of functions in* $L^2_w(I)$ *is* orthogonal *if:*

$$\langle \varphi_m, \varphi_n \rangle_w = 0 \qquad \text{for } m \neq n.$$

If moreover the linear subspace they span is dense in $L^2_w(I)$ *for the* $\|\cdot\|_{2,w}$ *norm, we say that this family is a* Hilbert basis *of* $L^2_w(I)$.

Fourier approximation theory

From the historical perspective, the Fourier theory of trigonometric sums is the first and still most popular example of L^2 approximation [134]. Let us consider $L^2([0, 2\pi])$ the space of measurable and square-integrable real-valued functions over $[0, 2\pi]$, or equivalently the 2π-periodic measurable functions over $\mathbb{R}$ with finite quadratic integral over one period (here, $w(x) = 1$). The linear space $L^2([0, 2\pi])$ equipped with the inner product $\langle f, g \rangle_{\mathcal{F}} = \int_0^{2\pi} f(t)g(t)\mathrm{d}t$ and the corresponding L^2-norm $\|f\|_{2,\mathcal{F}} = \int_0^{2\pi} f(t)^2 \mathrm{d}t$ is a Hilbert space. The Fourier theory proposes to approximate functions of $L^2([0, 2\pi])$ with trigonometric functions in:

$$\mathcal{A} = \bigcup_{n=0}^{\infty} \mathcal{A}_n \quad \text{where } \mathcal{A}_n = \mathrm{Span}_{\mathbb{R}} \{1, \cos t, \sin t, \ldots, \cos nt, \sin nt\}.$$

One easily checks that the elements of $\mathcal{A}$ form an orthogonal family with respect to $\langle \cdot, \cdot \rangle_{\mathcal{F}}$. The Fourier coefficients of $f \in L^2([0, 2\pi])$ are simply defined by orthogonal projection:

$$a_0(f) = \frac{\langle f, 1 \rangle_{\mathcal{F}}}{\|1\|^2_{2,\mathcal{F}}} = \frac{1}{2\pi} \int_0^{2\pi} f(t)\mathrm{d}t, \quad a_n(f) = \frac{\langle f, \cos nt \rangle_{\mathcal{F}}}{\|\cos nt\|^2_{2,\mathcal{F}}} = \frac{1}{\pi} \int_0^{2\pi} f(t) \cos nt \mathrm{d}t,$$

$$b_n(f) = \frac{\langle f, \sin nt \rangle_{\mathcal{F}}}{\|\sin nt\|^2_{2,\mathcal{F}}} = \frac{1}{\pi} \int_0^{2\pi} f(t) \sin nt \mathrm{d}t, \qquad \text{for all } n \geqslant 1.$$

This allows for computing the best L^2 by orthogonal projection. The very classical convergence properties of Fourier series can be found in **[134,** Chaps. I and II**]**. We omit them in the manuscript, since they do not play an important role in this thesis, contrary to the very related Chebyshev polynomials and series, presented in **Section 2.2.4**.

Orthogonal polynomials

The Fourier series presented above are useful to approximate periodic functions on the real line. However, for non-periodic functions, Fourier series are not well-behaved. According to the Weierstrass approximation theorem (**Theorem 2.19**), polynomials are natural candidates for the set $\mathcal{A}$ of approximating functions. We present here the notion of orthogonal polynomials, which can be seen as a generalization of Fourier approximation theory. More details can be found in **[53,** Chap. 4**]** or **[239]**.

Again, consider the space $L^2_w(I)$ of measurable and square integrable functions over the interval I (compact or not) with respect to the continuous and almost everywhere positive weight w. We moreover assume that all moments of w are finite:

$$\int_I x^n w(x)\mathrm{d}x < +\infty \qquad \text{for all } n \geqslant 0,$$

so that $L^2_w(I)$ contains all real-valued polynomials. We set $\mathcal{A} := \mathbb{R}[x]$.

Remark 2.25 *If $I = [a, b]$ is compact, then clearly $\mathbb{R}[x]$ is dense in $L^2_w(I)$ by using the Weierstrass approximation theorem and noticing that $\|f\|_{2,w} \leqslant (\int_I w(x)\mathrm{d}x)^{1/2}\|f\|_{\infty,I}$. Otherwise, one needs to perform a change of variable with the monotone $\mathcal{C}^1$ function $F(x) = \int_{-\infty}^x w(t)\mathrm{d}t$ of compact range, and use the Stone-Weierstrass theorem* **[53,** Sec. 6.1**]** *to deduce again the completeness of $\mathbb{R}[x]$ in $L^2_w(I)$.*

The completeness of $\mathbb{R}[x]$ in $L^2_w(I)$ raises the question of characterizing a graduated basis of orthogonal polynomials, forming a Hilbert basis of $L^2_w(I)$. Starting from the monomial basis $\{1, x, x^2, \dots, x^n, \dots\}$, the *Gram-Schmidt orthogonalization process* **[53,** Sec. 1.4**]** iteratively constructs such a graduated orthogonal family $P_0(x), P_1(x), P_2(x), \dots, P_n(x), \dots$ of monic polynomials:

- The procedure is initialized with $P_0(x) = 1$.
- Once $P_0(x), P_1(x), \dots, P_n(x)$ are constructed, we consider $xP_n(x) \in \mathbb{R}_{n+1}[x] \setminus \mathbb{R}_n[x]$. We have:
$$\langle xP_n(x), P_k(x)\rangle_w = \langle P_n(x), xP_k(x)\rangle_w = 0 \qquad \text{for } 0 \leqslant k < n-1,$$
since $P_n(x)$ is orthogonal to $\mathbb{R}_{n-1}[x]$. We therefore define:
$$P_{n+1}(x) = xP_n(x) - \frac{\langle xP_n(x), P_n(x)\rangle_w}{\|P_n(x)\|^2_{2,w}}P_n(x) - \frac{\langle xP_n(x), P_{n-1}(x)\rangle_w}{\|P_{n-1}(x)\|^2_{2,w}}P_{n-1}(x).$$
As required, $P_{n+1}(x)$ is monic of degree $n+1$ and orthogonal to $\mathbb{R}_n[x]$.

This leads to the following theorem **[53,** Sec 4.2, Thm. 2**]**:

Theorem 2.26 *Provided that the weight function w has finite moments on I, there exists a unique Hilbert basis of $L^2_w(I)$ made of graduated monic polynomials $P_0(x), P_1(x), \ldots, P_n(x), \ldots$. They satisfy a so-called three-term recurrence:*

$$P_{n+1}(x) = (x - \alpha_n)P_n(x) - \beta_n P_{n-1}(x), \qquad n \geqslant 1,$$

where the coefficients α_n, β_n are given by:

$$\alpha_n = \frac{\langle xP_n(x), P_n(x)\rangle_w}{\|P_n(x)\|^2_{2,w}}, \quad \beta_n = \frac{\langle xP_n(x), P_{n-1}(x)\rangle_w}{\|P_{n-1}(x)\|^2_{2,w}} = \frac{\|P_n(x)\|^2_{2,w}}{\|P_{n-1}(x)\|^2_{2,w}}.$$

Remark 2.27 *Clearly, the values of α_n and β_n depend on the weight function w. When they happen to be polynomials (or rational functions) of n, the sequence $(P_n(x))$ are* P-recursive *(see **Definition 2.5**). In particular, this is the case for Legendre and Chebyshev polynomials defined below.*

The orthogonal structure implies strong properties concerning the polynomials, like the location of the roots, characterized by next proposition **[53,** Sec. 4.2 Cor. 1**]**. In particular, this proves that the monomials $1, x, x^2, \ldots$ cannot be an orthogonal family over I, whatever the weight w is.

Proposition 2.28 *If $\{P_0, P_1, \ldots, P_n, \ldots\}$ is a graduated and orthogonal family of polynomials in $L^2_w(I)$, then P_n has exactly n distinct simple roots, distributed in the interior $\mathring{I}$ of I.*

Example 2.29 (Legendre polynomials) *By taking the weight $w(x) = 1$ over $I = [-1, 1]$, we obtain the simplest example of orthogonal polynomials: Legendre polynomials.*

$$P_n(x) = \frac{1}{2^n n!}\left(\frac{\partial}{\partial x}\right)^n \cdot \left[(x^2 - 1)^n\right].$$

*Moreover, the coefficients α_n and β_n of **Theorem 2.26** can be explicitly computed, yielding the so-called* Bonnet's recurrence*:*

$$(n + 1)P_{n+1}(x) = (2n + 1)xP_n(x) - nP_{n-1}(x).$$

2.2.4 ▶ Chebyshev polynomials and series

One of the most well-known family of orthogonal polynomials is the Chebyshev polynomials, due to the excellent approximation properties of Chebyshev expansions for functions defined over a compact interval. It is worth mentioning at this point that Chebyshev approximation theory is no more than Fourier approximation theory, up to a particular change of variable. Indeed, let $f, g : [-1, 1] \to \mathbb{R}$ two measurable functions, and consider the 2π-periodic functions $\tilde{f}(t) = f(\cos t)$, $\tilde{g}(t) = g(\cos t)$. We have:

$$\begin{aligned}
\langle \tilde{f}(t), \tilde{g}(t)\rangle_{\mathscr{F}} &= \int_0^{2\pi} f(\cos t)g(\cos t)\mathrm{d}t = 2\int_0^{\pi} f(\cos t)g(\cos t)\mathrm{d}t \\
&= 2\int_{-1}^{1} \frac{f(x)g(x)}{\sqrt{1 - x^2}}\mathrm{d}x \qquad \text{with } x = \cos t \\
&= 2\langle f(x), g(x)\rangle_w,
\end{aligned}$$

where $w(x) = \frac{1}{\sqrt{1-x^2}}$ is the *Chebyshev weight*. In the following, we write $L^2_{\text{Ч}} = L^2_w([-1,1])$, $\langle\cdot,\cdot\rangle_{\text{Ч}} = \langle\cdot,\cdot\rangle_w$ and $\|\cdot\|_{2,\text{Ч}} = \|\cdot\|_{2,w}$ when $w(x)$ is the Chebyshev weight defined on the interval $[-1,1]$.

Due to the central role played by Chebyshev polynomials throughout this thesis, a short summary of their properties is given in the following lines. For more details and proofs, I recommend these reference textbooks **[243, 34, 84, 170, 212]**.

Properties of Chebyshev polynomials

The image of $\mathbb{R}[x]$ under the change of variable $x = \cos t$ are the (pair) trigonometric polynomials, for which we already know an orthogonal basis $\{1, \cos t, \cos 2t, \dots, \cos nt, \dots\}$. We can therefore define the Chebyshev polynomials:

$$T_n(x) = \cos(n \arccos x) \qquad n \geqslant 0, \quad x \in [-1,1].$$

They clearly satisfy:

$$T_n(\cos t) = \cos nt \quad \text{for } n \geqslant 0, \qquad \text{and} \quad \langle T_n, T_m\rangle_{\text{Ч}} = 0 \quad \text{if} \quad n \neq m. \tag{2.4}$$

The polynomial T_n is of degree n with leading coefficient 2^{n-1} (for $n \geqslant 1$). The n distinct roots $\mu_k^{(n)}$ of T_n in $[-1,1]$, predicted by **Proposition 2.28**, are called the *Chebyshev nodes of the first kind* are, whereas the *Chebyshev nodes of the second kind* denote the $n+1$ local extrema $\nu_k^{(n)}$ of T_n. They are respectively given, in decreasing order, by:

$$\begin{aligned} \mu_k^{(n)} &= \cos\left(\frac{(k-1/2)\pi}{n}\right), && k \in [\![1,n]\!], \\ \nu_k^{(n)} &= \cos\left(\frac{k\pi}{n}\right), && k \in [\![0,n]\!]. \end{aligned}$$

The 3-term recurrence relation for orthogonal polynomials is here explicitly given using the trigonometric relation (2.4) (Note however that the T_n are not chosen monic by convention, whence the small difference with the statement of **Theorem 2.26**):

$$\begin{aligned} T_0(x) = 1, \qquad T_1(x) = x, \\ T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n \geqslant 1. \end{aligned} \tag{2.5}$$

Usual operations on Chebyshev polynomials also admit simple expressions:

$$T_n(x)T_m(x) = \frac{1}{2}\left(T_{n+m}(x) + T_{n-m}(x)\right), \qquad n \geqslant m, \tag{2.6}$$

$$\int T_n(x)\mathrm{d}x = \frac{1}{2}\left(\frac{T_{n+1}(x)}{n+1} - \frac{T_{n-1}(x)}{n-1}\right), \qquad n \geqslant 2, \tag{2.7}$$

$$T_n'(x) = \begin{cases} 2n(T_{n-1}(x) + T_{n-3}(x) + \cdots + T_1(x)), & n \text{ even}, \\ 2n(T_{n-1}(x) + T_{n-3}(x) + \cdots + T_2(x)) + nT_0(x), & n \text{ odd}. \end{cases} \tag{2.8}$$

Algorithm 2.2 CLENSHAW(p, x) – Clenshaw evaluation algorithm

Input: Polynomial $p = \sum_{i=0}^{n} a_i T_i$ in Chebyshev basis, and $x \in \mathbb{R}$.
Output: Evaluation $y = p(x)$ of p at x.

1: $b_{n+1} \leftarrow 0$ and $b_n \leftarrow a_n$
2: **for** $i = n-1$ **downto** 0 **do**
3: $\quad b_k \leftarrow a_k + 2x b_{k+1} - b_{k+2}$
4: **end for**
5: $y \leftarrow b_0 - x b_1$
6: **return** y

Clenshaw evaluation. A straightforward evaluation strategy for a polynomial $p = \sum_{i=1}^{n} a_i T_i$ at a point $x \in \mathbb{R}$ relies on the forward computation of the sequence $(T_i(x))_{0 \leqslant i \leqslant n}$, using the recurrence relation (2.5). However, the backward evaluation scheme due to Clenshaw [60] (**Algorithm** CLENSHAW) should be preferred in the general case, similarly to the Horner scheme in the monomial basis. Besides involving fewer arithmetic operations, it is more competitive regarding numerical stability, as the error analysis carried out in [84, §3.13] shows.

Proposition 2.30 (Correctness of CLENSHAW) *Under the hypothesis of exact arithmetic operations, **Algorithm** CLENSHAW(p, x) computes the expected evaluation $p(x)$ of p at x.*

Proof. For fixed $x \in \mathbb{R}$, we prove by decreasing induction from $k = n$ down to 0 the following equality:

$$(b_k - 2x b_{k+1}) T_k(x) + b_{k+1} T_{k+1}(x) = \sum_{i=k}^{n} a_i T_i(x). \qquad (P_k)$$

(P_n) is trivially true since $b_n = a_n$ and $b_{n+1} = 0$ (line 1). Now let $k \in [\![0, n-1]\!]$ and suppose that (P_{k+1}) holds. Using the Chebyshev recurrence relation (2.5), we obtain:

$$\sum_{i=k+1}^{n} a_i T_i(x) = (b_{k+1} - 2x b_{k+2}) T_{k+1}(x) + b_{k+2} T_{k+2}(x) = b_{k+1} T_{k+1} - b_{k+2} T_k(x).$$

Equality (P_k) easily follows from $b_k = a_k + 2x b_{k+1} - b_{k+2}$ (line 3):

$$(b_k - 2x b_{k+1}) T_k(x) + b_{k+1} T_{k+1}(x) = (a_k - b_{k+2}) T_k(x) + b_{k+1} T_{k+1}(x) = a_k T_k(x) + \sum_{i=k+1}^{n} a_i T_i(x).$$

This concludes the proof of (P_k) for $k \in [\![0, n]\!]$.

Finally, the correctness of **Algorithm** CLENSHAW follows from (P_0) and line 5:

$$p(x) = \sum_{i=0}^{n} a_i T_i(x) = (b_0 - 2x b_1) + b_1 x = b_0 - x b_1 = y.$$

□

Chebyshev series and convergence theorems

The Chebyshev coefficients $c_n(f)$ of f are defined by orthogonal projection:

$$\begin{aligned} c_0(f) &:= \frac{\langle f(x), 1\rangle_{\mathrm{T}}}{\|1\|^2_{2,\mathrm{T}}} = \frac{1}{\pi}\int_0^\pi f(\cos t)\mathrm{d}t, \\ c_n(f) &:= \frac{\langle f(x), T_n(x)\rangle_{\mathrm{T}}}{\|T_n(x)\|^2_{2,\mathrm{T}}} = \frac{2}{\pi}\int_0^\pi f(\cos t)\cos nt\mathrm{d}t, \qquad n \geqslant 1. \end{aligned} \tag{2.9}$$

As expected, we have $c_n(f) = a_n(f \circ \cos)$. The truncated Chebyshev series of f of degree n is defined by the orthogonal projection $\boldsymbol{\pi}_n : L^2_{\mathrm{T}} \to \mathbb{R}_n[x]$:

$$\boldsymbol{\pi}_n \cdot f(x) = \sum_{k=0}^{n} c_k(f) T_k(x).$$

Again, $(\boldsymbol{\pi}_n \cdot f) \circ \cos = \boldsymbol{\mathcal{F}}_n \cdot (f \circ \cos)$ for all $n \geqslant 0$.

The L^2-convergence theorem for Chebyshev series directly follows from Parseval theorem in the Fourier theory:

Theorem 2.31 (L^2 convergence of Chebyshev series) *For any $f \in L^2_{\mathrm{T}}$, we have:*

- $\boldsymbol{\pi}_n \cdot f \to f$ *in the L^2 sense when $n \to \infty$:*

$$\|f - \boldsymbol{\pi}_n \cdot f\|^2_{2,\mathrm{T}} = \int_{-1}^{1} \frac{(f(x) - \boldsymbol{\pi}_n \cdot f(x))^2}{\sqrt{1-x^2}}\mathrm{d}x \to_{n\to\infty} 0.$$

- Parseval identity*:*

$$\|f\|^2_{2,\mathrm{T}} = \pi\left(c_0(f)^2 + \frac{1}{4}\sum_{n\geqslant 1} c_n(f)^2\right).$$

Besides L^2 convergence, uniform convergence happens under very mild assumptions, for instance, Lipschitz continuity **[243,** Thm. 3.1**]**.

Theorem 2.32 *Let $f \in \mathcal{C}^0$ be Lipschitz-continuous, that is, there exists a $\lambda \geqslant 0$ such that $|f(x) - f(y)| \leqslant \lambda|x - y|$ for all $x, y \in [-1, 1]$. Then f has a unique representation as a Chebyshev series, which is absolutely and uniformly convergent:*

$$f(x) = \sum_{n=0}^{+\infty} c_n(f) T_n(x) \quad \textit{for all } x \in [-1,1], \qquad \textit{and} \quad \sum_{n=0}^{+\infty} |c_n(f)| < \infty.$$

Similarly to the Fourier case, the rate of convergence of Chebyshev coefficients for a function f depends on the regularity of the latter. This allows for bounding the *uniform* approximation error of Chebyshev series **[243,** Thms. 7.1 and 7.2**]**.

Theorem 2.33 (Convergence of Chebyshev series for differentiable functions) *Let $p \geqslant 0$ and f be a p times differentiable function over $[-1, 1]$, with*

$$L_{(p)} := \int_{-1}^{1} |f^{(p)}(x)|\mathrm{d}x < +\infty.$$

(i) The sequence of Chebyshev coefficients $c_n(f)$ converge to 0 in $O(n^{-p})$, with the explicit bound:

$$|c_n(f)| \leqslant \frac{2L_{(p)}}{\pi(n-p+1)^p}, \qquad \text{for } n \geqslant p.$$

(ii) If $p \geqslant 1$, the truncated Chebyshev series $\mathbf{\pi}_n \cdot f$ converges to f uniformly and absolutely in $\mathcal{O}(n^{1-p})$:

$$\|f - \mathbf{\pi}_n \cdot f\|_\infty \leqslant \frac{2L_{(p)}}{\pi(p-1)(n-p+1)^{p-1}}, \qquad \text{for } n \geqslant p.$$

Hence, contrary to Taylor series, Chebyshev series can be defined even for non analytic functions, and convergence happens as soon as $f \in \mathcal{C}^1([-1,1])$. For the sake of completeness, we also cite this Taylor-like theorem for Chebyshev series [78]. Notice the improvement of a factor 2^{-n} compared to the Taylor case.

Theorem 2.34 *If $f \in \mathcal{C}^{n+1}([-1,1])$, then there exists $\xi \in [-1,1]$ such that:*

$$\|f - \mathbf{\pi}_n \cdot f\|_\infty \leqslant \frac{|f^{(n+1)}(\xi)|}{2^n n!}.$$

Domain of analytic convergence

When f happens to be analytic on some neighborhood of $[-1,1]$ in the complex plane, Cauchy-like results for Chebyshev series provide a geometric convergence rate.

Remember that for Taylor series (at 0) for a function f analytic around 0, the convergence domain is the largest open disc centered at 0 and avoiding all singularities of f. Hence, when f has complex singularities of modulus less than 1, its Taylor series fails to converge on $[-1,1]$ (see **Figure 2.2a**).

To investigate the Chebyshev case, we use the *Joukowski transform* [243, Chap. 8] $x = (z + z^{-1})/2$ and the following identity, which generalizes the trigonometric relation (2.4) over $\mathbb{C}^*$:

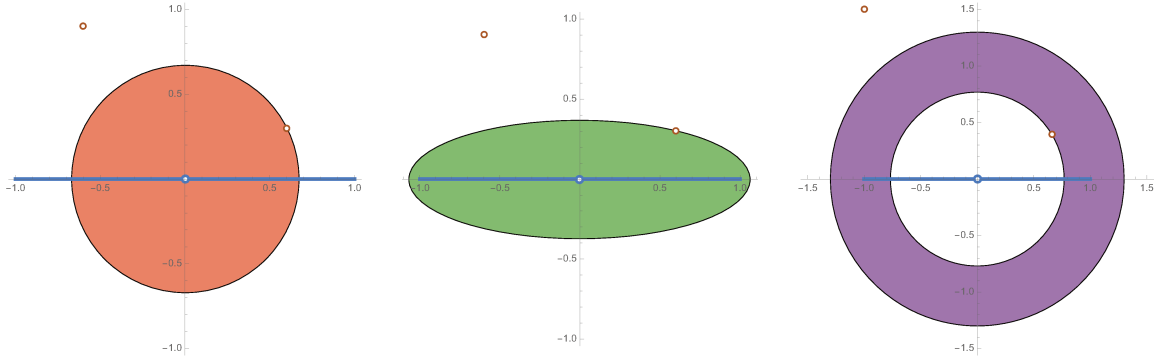
$$T_n\left(\frac{z+z^{-1}}{2}\right) = \frac{z^n + z^{-n}}{2}, \qquad z \in \mathbb{C}^*, \quad n \geqslant 0.$$

The segment $[-1,1]$ is mapped to the unit circle $\mathcal{U} = \{z \in \mathbb{C} \mid |z| = 1\}$, and a Chebyshev series $\sum_{n\geqslant 0} a_n T_n(x)$ to $a_0 + \frac{1}{2}\sum_{n\in\mathbb{Z}^*} a_{|n|} z^n$, called a Laurent series. The domain of convergence for a Laurent series is an *annulus* $\mathcal{C}_{R,r} = \{z \in \mathbb{C} \mid r < |z| < R\}$ of inner and outer radius $0 < r < R < +\infty$. Since the Laurent series obtained from real Chebyshev series are invariant under $z \mapsto z^{-1}$, the domains of convergence we consider are the symmetric annuli $\mathcal{C}_{\rho,\rho^{-1}}$ for some $\rho > 1$ (see **Figure 2.2c**). The Joukowski transform maps the annulus $\mathcal{C}_{\rho,\rho^{-1}}$ to the Bernstein ellipse of "radius" $\rho > 1$ (see **Figure 2.2b**):

$$\mathcal{E}_\rho = \{x \in \mathbb{C} \mid |x + \sqrt{x^2-1}| < \rho\}.$$

The key advantage of Chebyshev series over Taylor series is that, whatever close to $[-1,1]$ the complex singularities of f are, there always exists a sufficiently small $\rho > 1$ such that $\mathcal{E}_\rho$ avoids them.

A generalization of the Cauchy formula for the coefficients of a Laurent series leads to the following theorem establishing the geometric convergence rate of Chebyshev coefficients and Chebyshev series [243, Thms. 8.1 and 8.2].



(a) Disk for Taylor series

(b) Bernstein ellipse for Chebyshev series

(c) Annulus for Laurent series

Figure 2.2: Typical domains of convergence for various series expansions, limited by the closest singularities in the complex plane (in orange on the plot).

Theorem 2.35 (Convergence of Chebyshev series for analytic functions) *Let f be a smooth function on $[-1,1]$ that admits an analytic continuation on the open Bernstein ellipse $\mathcal{E}_\rho$ for some $\rho > 1$. Suppose moreover that $|f(x)| \leqslant M$ for all $x \in \mathcal{E}_\rho$, for some M. Then:*

(i) The sequence of Chebyshev coefficients of f decays geometrically:

$$|c_0(f)| \leqslant M, \qquad \text{and} \qquad |c_n(f)| \leqslant 2M\rho^{-n}, \quad n \geqslant 1.$$

(ii) The truncated Chebyshev series converge geometrically fast to f on $[-1,1]$:

$$\|f - \boldsymbol{\pi}_n \cdot f\|_\infty \leqslant \frac{2M\rho^{-n}}{\rho - 1}, \qquad n \geqslant 0.$$

Chebyshev series are near-best approximations

Theorems 2.33 and **2.35** describe how the truncated Chebyshev series $\boldsymbol{\pi}_n \cdot f$ converges to the function f, depending on regularity assumptions on f. Yet, for a fixed degree n, this does not characterize how far this approximation error is from the optimal one given by the minimax approximation p_n^* of f. As it was seen previously, the Lebesgue constant $\Lambda_n^{\boldsymbol{\pi}}$ associated to $\boldsymbol{\pi}_n$ is the appropriate quantity to understand this approximation defect (see **[243,** Thm. 15.3**]**).

Proposition 2.36 *The Lebesgue constant $\Lambda_n^{\boldsymbol{\pi}}$ associated to the orthogonal projection operator $\boldsymbol{\pi}_n$ is given by:*

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

with the following bound and asymptotic:

$$\Lambda_n^{\boldsymbol{\pi}} \leqslant \frac{4}{\pi^2} \log(n+1) + 3 \qquad \text{and} \qquad \Lambda_n^{\boldsymbol{\pi}} \sim \frac{4}{\pi^2} \log n, \quad n \to +\infty.$$

This means that using truncated Chebyshev series instead of the minimax polynomial induces an approximation defect that grows *slowly* with the degree n, whence the denomination *near-best approximations*. Considering the practical difficulty of computing minimax polynomials, this emphasizes the role of Chebyshev series in approximation theory.

The Banach space $(\mathrm{Ч}^1, \|\cdot\|_{\mathrm{Ч}^1})$

It turns out that working directly with the $\|\cdot\|_\infty$ norm may be cumbersome in effective validated algorithms. In some situations, as in **Chapters 3**, **4** and **5**, it is more convenient to use a subspace of $\mathcal{C}^0$ and an appropriate norm in connection with Chebyshev coefficients, while not loosing the link with $\|\cdot\|_\infty$. This leads to the Banach space $(\mathrm{Ч}^1, \|\cdot\|_{\mathrm{Ч}^1})$, defined below with relevant properties.

Definition 2.37 *$\mathrm{Ч}^1$ is the space of absolutely summable Chebyshev series, that is, functions $f \in L^2_{\mathrm{Ч}}$ such that $\|f\|_{\mathrm{Ч}^1} < +\infty$, where:*

$$\|f\|_{\mathrm{Ч}^1} := \sum_{n=0}^{\infty} |c_n(f)| \in [0, +\infty].$$

These functions exactly coincide with their Chebyshev series in the following sense:

Lemma 2.38 *If $f \in \mathrm{Ч}^1$, then $\mathbf{п}_n \cdot f$ converges absolutely and uniformly to f.*

Proof. Since, for all $i \in \mathbb{N}$, $\|c_i(f)T_i\|_\infty \leqslant |c_i(f)|$ and $\sum_{i=0}^{\infty} |c_i(f)| = \|f\|_{\mathrm{Ч}^1} < \infty$ by definition of $f \in \mathrm{Ч}^1$, $\mathbf{п}_n \cdot f := \sum_{i=0}^{n} c_i(f)T_i$ converges absolutely and uniformly to a continuous function $\widehat{f}$, and therefore also in L^2. But since $\mathbf{п}_n \cdot f \to f$ in L^2 (by **Theorem 2.31**), we have $f = \widehat{f}$ almost everywhere, and in fact everywhere, by continuity. □

Note that $\mathrm{Ч}^1$ is analogous to the Wiener algebra $A(\mathbb{T})$ of absolutely convergent Fourier series **[134, §I.6]**: for $f \in \mathrm{Ч}^1$, we have $\|f\|_{\mathrm{Ч}^1} = \|f(\cos)\|_{A(\mathbb{T})}$. More precisely we have:

Lemma 2.39 *$(\mathrm{Ч}^1, \|\cdot\|_{\mathrm{Ч}^1})$ is a Banach algebra, which means that it is a Banach space satisfying*

$$\|fg\|_{\mathrm{Ч}^1} \leqslant \|f\|_{\mathrm{Ч}^1}\|g\|_{\mathrm{Ч}^1} \quad \text{for all } f, g \in \mathrm{Ч}^1. \tag{2.10}$$

Proof. It is identical to the proofs from **[134, §I.6]**. □

It follows from **Lemma 2.38** and **Theorem 2.32** that $\mathrm{Ч}^1$ is included in $\mathcal{C}^0$ and contains the set of Lipschitz functions over $[-1, 1]$. Actually, the inclusions are strict, see **[271, §VIII.1]** and **[271, §VI.3]** respectively.

Moreover, the uniform and $\mathrm{Ч}^1$ norms can be partially ordered:

$$\|g\|_\infty \leqslant \sum_{n=0}^{\infty} \|c_n(g)T_n\|_\infty \leqslant \sum_{n=0}^{\infty} |c_n(g)| =: \|g\|_{\mathrm{Ч}^1} \quad \text{for all } g \in \mathrm{Ч}^1.$$

Conversely, we have from **(2.9)**:

$$|c_0(f)| \leqslant \|f\|_\infty \quad \text{and} \quad |c_n(f)| \leqslant 2\|f\|_\infty, \qquad \text{for all} \quad n \geqslant 1, \quad f \in \mathrm{Ч}^1.$$

However, since f has in general an infinite number of non-zero coefficients, this fact cannot be used directly to bound $\|f\|_{\mathrm{Ч}^1}$ by the uniform norm of f.

We now consider the action of a bounded linear operator $\mathbf{F} : \mathrm{Ч}^1 \to \mathrm{Ч}^1$. By definition, its operator norm is $\|\mathbf{F}\|_{\mathrm{Ч}^1} := \sup_{\|f\|_{\mathrm{Ч}^1} \leqslant 1} \|\mathbf{F} \cdot f\|_{\mathrm{Ч}^1}$. Such operators include multiplication by $f \in \mathrm{Ч}^1$ or integration (indefinite or from a specific point).

Proposition 2.40 *Let $f = \sum_{n=0}^{+\infty} a_n T_n \in \text{Ч}^1$. For indefinite integration operator $\int$ and respectively definite integration $\int_{t_0}^{t}$ from specific $t_0 \in [-1, 1]$, defined as:*

$$\begin{aligned} \int f &:= \left(a_0 + \frac{a_2}{2}\right) T_1 + \sum_{n=2}^{+\infty} \frac{a_{n-1} - a_{n+1}}{2n} T_n, \\ \int_{t_0}^{t} f \mathrm{d}t &:= \left(a_0 + \frac{a_2}{2}\right)(t - t_0) + \sum_{n=2}^{+\infty} \frac{a_{n-1} - a_{n+1}}{2n} (T_n(t) - T_n(t_0)), \end{aligned} \tag{2.11}$$

we have the following Ч^1-operator norms:

$$\left\| \int \right\|_{\text{Ч}^1} = 1, \qquad \text{and} \qquad \left\| \int_{t_0}^{t} \right\|_{\text{Ч}^1} \leqslant 2. \tag{2.12}$$

Proof. The inequality $\| \int \|_{\text{Ч}^1} \leqslant 1$ directly follows from the definition, and equality is attained with $\int T_0 = T_1$. For definite integration the operator bound is tight for $t_0 = -1$ since $\| \int_{-1}^{t} T_0 \mathrm{d}t \|_{\text{Ч}^1} = \|T_1 + T_0\|_{\text{Ч}^1} = 2$, but not for $t_0 = 0$, where $\| \int_0^t \|_{\text{Ч}^1} = 1$. □

In general, computing the Ч^1-norm of an operator $\mathbf{F}$ reduces to evaluating $\mathbf{F}$ at all the polynomials T_i for $i \in \mathbb{N}$:

Lemma 2.41 *For a bounded linear operator $\mathbf{F} : \text{Ч}^1 \to \text{Ч}^1$, its Ч^1-operator norm is given by:*

$$\|\mathbf{F}\|_{\text{Ч}^1} = \sup_{i \geqslant 0} \|\mathbf{F} \cdot T_i\|_{\text{Ч}^1}.$$

Proof. Take $f = \sum_{n=0}^{+\infty} a_n T_n \in \text{Ч}^1$. We have:

$$\begin{aligned} \|\mathbf{F} \cdot f\|_{\text{Ч}^1} &= \left\| \sum_{n=0}^{+\infty} a_n \mathbf{F} \cdot T_n \right\|_{\text{Ч}^1} \leqslant \sum_{n=0}^{+\infty} |a_n| \|\mathbf{F} \cdot T_n\|_{\text{Ч}^1} \\ &\leqslant \left(\sum_{n=0}^{+\infty} |a_n| \right) \sup_{i \geqslant 0} \|\mathbf{F} \cdot T_i\|_{\text{Ч}^1} = \|f\|_{\text{Ч}^1} \sup_{i \geqslant 0} \|\mathbf{F} \cdot T_i\|_{\text{Ч}^1} \end{aligned}$$

which shows that $\|\mathbf{F}\|_{\text{Ч}^1} \leqslant \sup_{i \geqslant 0} \|\mathbf{F} \cdot T_i\|_{\text{Ч}^1}$. The converse inequality is clearly true since the family of the T_i is a subset of $\{f \in \text{Ч}^1 \mid \|f\|_{\text{Ч}^1} \leqslant 1\}$. □

By analogy with the Wiener algebra $A(\mathbb{T})$, more results about the space Ч^1 could be given. For example, an interesting property is the closure under the reciprocal: if $f \in \text{Ч}^1$ does not vanish over $[-1, 1]$, then $1/f \in \text{Ч}^1$ **[134,** Sec. VIII.6.1**]**.

2.2.5 ▶ Chebyshev interpolation

Despite their excellent approximation properties, the Chebyshev series described above may be difficult to obtain. Indeed, with no specific assumptions on the function to approximate,

computing Chebyshev coefficients amounts to evaluating integrals, which is not particularly convenient. An alternative consists in computing *Chebyshev interpolants*, whose properties of *near-best approximations* will be given after a general introduction on polynomial interpolation.

Let I be an interval of the real line, and $n+1$ points $x_0 < x_1 < x_2 < \cdots < x_n$ in I. For given values $y_1, \ldots, y_n \in \mathbb{R}$, there exists a unique polynomial $p \in \mathbb{R}_n[x]$ such that $p(x_i) = y_i$ for $i \in [\![0, n]\!]$. For $i \in [\![0, n]\!]$, let $l_i(x)$ denote the degree n polynomial satisfying $l_i(x_i) = 1$ and $l_i(x_j) = 0$ for $j \neq i$ (see expression below). We define the linear projection $\boldsymbol{\mathcal{I}}_n$ that *interpolates* the continuous function f at the points x_i:

$$\begin{array}{rccl} \boldsymbol{\mathcal{I}}_n & : & \mathcal{C}^0(I) & \to & \mathbb{R}_n[x] \\ & & f & \mapsto & \sum_{i=0}^n f(x_i) l_i(x) \quad \text{with} \quad l_i(x) = \displaystyle\prod_{\substack{0 \leqslant j \leqslant n \\ j \neq i}} \frac{x - x_j}{x_i - x_j}. \end{array}$$

Remark 2.42 *Several methods are available to implement polynomial interpolation, refered to as Lagrange, Newton interpolation formulas (see, e.g.,* **[53,** Sec. 3.2**]**).

The approximation error of polynomial interpolants can be expressed using a Taylor-Lagrange-like formula, by a repeated use of the Rolle theorem **[53,** Sec. 3.2 p. 60**]**.

Theorem 2.43 *Let $n \geqslant 0$, $f \in \mathcal{C}^{(n+1)}(I)$, $\boldsymbol{\mathcal{I}}_n \cdot f$ its degree n interpolant at points x_i in I, and $W(x) = \prod_{i=0}^n (x - x_i)$. For any $x \in I$, there exists $\xi \in I$ such that:*

$$f(x) - \boldsymbol{\mathcal{I}}_n \cdot f(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} W(x).$$

This theorem motivates the need of minimizing $\|W\|_{\infty, I}$ when $I = [a, b]$ is compact. Without loss of generality, we may assume $I = [-1, 1]$. The polynomial $W(x)$ is monic of degree $n+1$, with $n+1$ simple roots in $[-1, 1]$. By writing $W(x) = x^{n+1} - \sum_{i=0}^n a_n x^n$, we can see it as the approximation error of x^{n+1} by a degree n polynomial, and the best uniform bound is given by the minimax polynomial. Now, $2^{-n} T_{n+1}(x)$ is a monic degree $n+1$ polynomial with all its roots in $[-1, 1]$, and its local extrema $\nu_k^{(n+1)}$ for $k \in [\![0, n+1]\!]$ provide the equioscillation condition of **Theorem 2.22**. Hence, $2^{-n} T_{n+1}(x) = \prod_{i=1}^{n+1} (x - \mu_i^{(n+1)})$ is the optimal choice for $W(x)$. We therefore focus on *Chebyshev interpolation of the first kind*, at points $x_i = \mu_i^{(n+1)} = \cos((i - 1/2)\pi/(n+1))$, $i \in [\![1, n+1]\!]$. We call $\tilde{\boldsymbol{\pi}}_n$ the associated interpolation operator. In the following, we set $I = [-1, 1]$ and write $\|\cdot\|_\infty$ for $\|\cdot\|_{\infty, [-1,1]}$.

Theorem 2.44 *Let $n \geqslant 0$, and $f \in \mathcal{C}^{(n+1)}([-1, 1])$. The approximation error of f by its degree n Chebyshev interpolant $\tilde{\boldsymbol{\pi}}_n \cdot f$ is bounded by:*

$$\|f - \tilde{\boldsymbol{\pi}}_n \cdot f\|_\infty \leqslant \frac{\|f^{(n+1)}\|_\infty}{2^n (n+1)!}.$$

Chebyshev interpolants are near-best approximations

Similarly to Chebyshev expansions, The Lebesgue constant $\Lambda_n^{\tilde{\boldsymbol{\pi}}}$ for Chebyshev interpolation grows logarithmically with the degree n **[243,** Chap. 15**]**, **[264]**.

Proposition 2.45 *The Lebesgue constant $\Lambda_n^{\tilde{\mathbf{\Pi}}}$ associated to the Chebyshev interpolation operator $\tilde{\mathbf{\Pi}}_n$ is given by:*

$$\Lambda_n^{\tilde{\mathbf{\Pi}}} = \frac{1}{\pi} \sum_{k=1}^{n+1} \left| \operatorname{cotan} \frac{(k-1/2)\pi}{2(n+1)} \right|,$$

with the following bound and asymptotic:

$$\Lambda_n^{\tilde{\mathbf{\Pi}}} \leqslant \frac{2}{\pi} \log(n+1) + 1 \qquad \textit{and} \qquad \Lambda_n^{\tilde{\mathbf{\Pi}}} \sim \frac{2}{\pi} \log n, \quad n \to +\infty.$$

Convergence properties of Chebyshev interpolants

A key tool to investigate the convergence of Chebyshev interpolants is the *aliasing formula* (see **[243,** Chap. 4**]**). Roughly speaking, this formula establishes a partition of all the T_k by their contribution on the $n+1$-points Chebyshev grid $\{\mu_i^{(n+1)}, 1 \leqslant i \leqslant n+1\}$. For a function f admitting an absolutely summable Chebyshev series, this allows us to relate the $n+1$ coefficients of the interpolant $\tilde{\mathbf{\Pi}}_n \cdot f$ with the Chebyshev coefficients of f. The following consequence will be useful to establish convergence properties of Chebyshev interpolants.

Proposition 2.46 (**[243,** Eq. (4.9)**]**) *Let $f \in \mathcal{C}^0([-1,1])$ of absolutely summable Chebyshev series, and an interpolation degree n. Then:*

$$\|f - \tilde{\mathbf{\Pi}}_n \cdot f\|_\infty \leqslant 2 \sum_{k=n+1}^{+\infty} |c_k(f)|.$$

In particular, $\tilde{\mathbf{\Pi}}_n \cdot f$ converges uniformly to f.

The bound given by this theorem implies that all convergence theorems for Chebyshev series based on term-by-term bounds are true for Chebyshev interpolants, up to a factor of 2.

Theorem 2.47 *Let f be continuous on $[-1,1]$ with an absolutely summable Chebyshev series.*

(i) If $f \in \mathcal{C}^p([-1,1])$ for $p \geqslant 1$ with $L_{(p)} = \int_{-1}^{1} |f^{(p)}(x)| \mathrm{d}x$, then:

$$\|f - \tilde{\mathbf{\Pi}}_n \cdot f\|_\infty \leqslant \frac{4 L_{(p)}}{\pi (p-1)(n-p+1)^{p-1}}, \qquad n \geqslant p.$$

(ii) If f admits an analytic continuation on the Bernstein ellipse $\mathcal{E}_\rho$ for some $\rho > 1$, such that $|f(x)| \leqslant M$ for all $x \in \mathcal{E}_\rho$, for some M, then:

$$\|f - \tilde{\mathbf{\Pi}}_n \cdot f\|_\infty \leqslant \frac{4 M \rho^{-n}}{\rho - 1}, \qquad n \geqslant 0.$$

Discrete cosine transform (DCT)

Discrete cosine transform (DCT) designates the reciprocal of Chebyshev interpolation, that is, the evaluation of a polynomial $p = \sum_{i=0}^{n} a_i T_i$ over the Chebyshev grid of the first kind $(\mu_k^{(n+1)})_{k=1}^{n+1}$ or second kind $(\nu_k^{(n)})_{k=0}^{n}$. The naive method consists in applying the Clenshaw evaluation scheme (**Algorithm** Clenshaw) on each of the $n+1$ points, which requires a quadratic number of arithmetic operations.

Similarly to the Fast Fourier transform (FFT) in monomial basis, there exists a *fast cosine transform*, sometimes abusively called discrete cosine transform (which formally refers to the above mathematical operation, not how it is performed). More details can be found in **[198, 235]**. In particular, this allows for Chebyshev interpolation, multiplication of polynomials, and Chebyshev-to-monomial or monomial-to-Chebyshev change of basis, in quasi-linear time.

However, as it will be mentioned in the following chapter, designing a rigorous DCT is still ongoing research. Therefore, since we mainly focus on rigorous numerics in this manuscript, quasi-linear time DCT will not be used in the algorithms presented in the following chapters.

Rigorous Polynomial Approximations

3

Les propositions mathématiques sont reçues comme vraies parce que personne n'a intérêt qu'elles soient fausses.

— Montesquieu, Mes Pensées

In the light of the observations in **Chapter 1** that interval arithmetic is intrinsically limited by the loss of correlations between variables, the need for *higher order methods* became central in rigorous numerics. It is beyond the scope of this thesis to present all of them, and we instead refer the interested reader to more comprehensive references **[174, 247, 221]** to discover some of them, such as affine arithmetic or automatic differentiation. This chapter focuses on *rigorous polynomial approximations*, which are particularly relevant to tackle function space problems.

The key idea of rigorous polynomial approximations, often shortened as RPAs, is to use (polynomial) approximation theory, whose basic notions have been given in **Section 2.2**, in order to provide certified representations of functions. More specifically, let us remember that interval arithmetic uses floating-point numbers to build *sets* of real numbers, and rigorous operations guarantee that the exact mathematical result is always contained in the computed interval. Similarly, RPAs use polynomial approximations and rigorous error bounds to denote *sets of functions*, and operations on them must ensure that the resulting RPA always contains the exact function.

After an introductory discussion on the genesis of RPAs and related definition issues in **Section 3.1**, *self-validating* elementary operations on them are given in **Section 3.2**, meaning that, for instance, RPAs for addition and multiplication are directly constructed from RPAs of the operands, using mainly interval arithmetic operations. Although quite classical now in the RPA literature (see, e.g., the reference document **[129]** for Chebyshev basis), they are recalled in this chapter for the sake of completeness.

In contrast with the dominant trend in that topic, where more "complex" operations like division or square root of RPAs are handled via composition, we advocate in this chapter the use of *a posteriori* validation methods to provide more efficient algorithms. The general framework for *a posteriori* validation, relying on the Banach fixed-point theorem, is presented in **Section 3.3**. After that, the division and square root of RPAs are carried out in **Section 3.4**.

This chapter also gives me the opportunity to present two implementations, which are part of my contribution:

- An open source C library, named ChebValid, for RPAs in Chebyshev basis (a.k.a. Chebyshev models), available at `https://gforge.inria.fr/projects/tchebyapprox/`, implements all the elementary operations described in this chapter. The resulting collection of routines provides the necessary arithmetic on RPAs for the implementation of the validation method for LODEs in **Chapters 4** and **5**.

- An open source formalization in the Coq proof assistant, resulting from a joint work with Assia Mahboubi and Damien Pous, provides a framework for RPAs certified at the implementation level. Besides implementing the elementary operations of **Section 3.2**, we also give a formalization of the Banach fixed-point theorem in **Section 3.3.2** and use it to build certified implementations of the division and square root of RPAs, in **Section 3.4**. The Coq development is available at `http://perso.ens-lyon.fr/florent.brehard/chebapprox/`, and an article **[40]** – from which part of the material of this chapter comes – has been recently submitted to the *2019 Interactive Theorem Proving* conference.

Finally, examples illustrating these concepts and the two implementations are given in **Section 3.5**.

3.1 History and definition(s) of rigorous polynomial approximations

The idea of using *sets of functions* in computer programs to rigorously enclose the exact solution of function space problems originally appeared in **[79, 80]**, under the name of *ultra-arithmetic*, thus extending interval arithmetic, previously formalized in 1966 **[173]**. An early example of practical use of RPAs is the first computer-assisted proof of the Feigenbaum conjecture **[147]**. One decade later, the generic implementation of the so-called *Taylor models* **[164, 165]** in the COSY Infinity software for beam physics **[166]** enhanced the popularity of RPAs. Since then, Taylor models based methods have been used in various areas, such as rigorous integration of ODEs **[23, 163, 167, 181]** or rigorous range enclosures of functions **[24, 54]**. Further historical information and applications can be found in **[185,** Part 1**]**.

The central idea of RPAs consists in representing a function f over a compact segment I using a couple $\mathbb{f} := (p, \mathbb{\delta})$, called a Taylor model, where the approximation p is a polynomial expressed in the standard monomial basis, and the remainder $\mathbb{\delta}$ is an interval, such that for every $t \in I$, $f(t) - p(t) \in \mathbb{\delta}$. Despite its apparent simplicity, this definition requires some important clarifications when targeting practical implementations.

Floating-point vs interval coefficients: Rigorous operations on RPAs must take into account the rounding errors occuring when manipulating the polynomial approximations. One possible option **[164,** Chap. 5**]** is to use polynomials with floating-point coefficients, bound all the rounding errors and add them to the remainder $\mathbb{\delta}$. An alternative is to use *interval polynomials*, that is, polynomials $\mathbb{p}$ with interval coefficients, and use self-validating operations on them **[129,** Chap. 2**]**. In the latter case, which will be mainly

considered in this thesis, a precise semantic for the RPA $\mathbb{f} = (\mathbb{p}, \mathbb{δ})$ must be given (see **Definition 3.1** below).

Base interval: Is the interval $I = [a, b]$ fixed (we will often consider $I = [-1, 1]$ for Chebyshev models), or user-defined, e.g., given by a concrete interval $\mathbb{i} \in \mathbb{I}_{\mathbb{F}}$ or by two intervals $\mathbb{a}, \mathbb{b}$ for its endpoints?

Taylor coefficients: When using Taylor models with interval polynomials, we can impose that the coefficients of $\mathbb{p}$ contain the corresponding exact Taylor coefficients of the function to be represented. This can be useful when working with function with *singularities*, but this condition is not desirable in general, since maintaining it may be a cumbersome task **[129,** Chap. 2**]**.

In view of this, a salutary effort toward clarification has been made in **[129]**, leading for instance to the following definition of *Taylor models with absolute remainder* **[129,** Def. 2.1.3**]**.

Definition 3.1 *Let $f : I \to \mathbb{R}$ be a function, $\mathbb{t}_0$ be a small interval around an expansion point t_0. Let $\mathbb{f} = ((\mathbb{a}_0, \dots, \mathbb{a}_n), \mathbb{δ})$ be an RPA structure. We say that $\mathbb{f}$ is a Taylor model with absolute remainder for f at $\mathbb{t}_0$ on I if:*

$$\mathbb{t}_0 \subseteq I, \qquad 0 \in \mathbb{δ}, \qquad \text{and}$$

$$\forall \xi_0 \in \mathbb{t}_0,\ \exists_{0 \leqslant i \leqslant n} a_i \in \mathbb{a}_i,\ \forall t \in I,\ \exists \delta \in \mathbb{δ},\ f(t) - \sum_{i=0}^{n} a_i (t - \xi_0)^i = \delta.$$

However, the notion of RPA can be made broader than that of Taylor models, as shown by these two directions of generalization below.

Choice of the norm. Until now, we have only considered the space $\mathcal{C}^0(I)$ of continuous functions over a given compact segment I, equipped with the norm $\|\cdot\|_\infty$ of uniform convergence:

$$\|f\|_\infty := \sup_{t \in I} |f(t)|.$$

However, as it has been addressed in **Section 2.2**, other norms, like L^2 norms, can be used in approximation theory, and can be more relevant, depending on the context. Another example is the $\mathcal{Y}^1$-norm for absolutely convergent Chebyshev series (**Definition 2.37**). It will be useful in **Chapter 4** when computing RPAs for solutions of linear ODEs, and moreover this norm is a safe overapproximation of $\|\cdot\|_\infty$ over $[-1, 1]$, thus not losing the link with classical uniform convergence.

Therefore, from the *abstract* point of view, a RPA in a Banach space $(\mathcal{F}, \|\cdot\|)$ is simply a pair $\mathbb{f} := (f^\circ, \varepsilon)$ with f° an *approximation* lying in some suitable linear subspace $\mathcal{A}$ of $\mathcal{F}$ (most of the time but not necessarily polynomials) and $\varepsilon \geqslant 0$ an error bound.

Just like an interval is a *set* of real numbers, $\mathbb{f}$ represents a set of functions, namely the ball of center f° and radius ε. By a slight abuse of notation, we may use the membership symbol to say that $\mathbb{f}$ is a RPA for the target function $f^* \in \mathcal{F}$:

$$f^* \in \mathbb{f} \qquad \Leftrightarrow \qquad \|f^* - f^\circ\| \leqslant \varepsilon.$$

Choice of the basis. The name "rigorous polynomial approximation" suggests that the approximation set $\mathcal{A}$ should be $\mathbb{R}[t]$. This is the most common choice, but other choices are possible, like Bessel or Hermite functions. Even for a fixed $\mathcal{A}$, the choice of the basis in which the approximations are computed is crucial. Taylor models are based on (shifted) monomial basis. A major contribution of [44],[129, Chap. 4] is the use of Chebyshev series and interpolants, due to excellent convergent properties in general, summarized in Section 2.2. Such RPAs (Chebyshev basis and uniform norm) are sometimes referred to as *Chebyshev models*. They are the main target of this chapter, although a wide range of results and algorithms apply to other bases and norms.

While choosing (rescaled) Chebyshev basis for a compact $I = [a, b]$ is almost always a good option, other bases are more attractive when working with unbounded intervals [34, Chap. 17], like for instance Laguerre for $I = [0, +\infty)$, or Hermite for $I = (-\infty, +\infty)$. Obviously, in such situations, appropriate norms must be considered, such as (weighted) L^2 norms. Further investigations are needed in that direction. In the following, we will mainly consider the Chebyshev basis over a compact segment, say $I = [-1, 1]$ to avoid rescaling details.

In view of the preceding discussion, we propose the following definition of *rigorous polynomial approximations* for the rest of the thesis.

Definition 3.2 *Let $(\mathcal{F}, \|\cdot\|)$ be a Banach space of functions, $\mathcal{A} \subseteq \mathcal{F}$ be a linear subspace of approximating functions, abusively called* polynomials *and generated by a countable basis $(l_n)_{n\geqslant 0}$. An RPA $\mathbb{f}$ is a pair $((\mathbb{c}_0, \ldots, \mathbb{c}_n), \varepsilon)$, where:*

- *$(\mathbb{c}_0, \ldots, \mathbb{c}_n)$ is a finite sequence of (floating-point) intervals, representing an* interval polynomial $\mathbb{p} := \sum_{i=0}^{n} \mathbb{c}_i \ell_i$, *for which $p := \sum_{i=0}^{n} a_i \ell_i \in \mathbb{p}$ means $a_i \in \mathbb{c}_i$ for all $0 \leqslant i \leqslant n$;*
- *$\varepsilon \geqslant 0$ is an error bound, given as a floating-point number.*

An RPA $\mathbb{f}$ is said to represent a function $f^* \in \mathcal{F}$, which we write $f^* \in \mathbb{f}$, if and only if:

$$\exists p \in \mathbb{p}, \quad \|f^* - p\| \leqslant \varepsilon.$$

Implementation of RPAs in C

The ChebValid C library only implements Chebyshev models for now. They are defined according to the polynomial with interval coefficients – error bound approach given above.

The C code follows the GMP/MPFR/MPFI naming convention for functions: all operations concerning Chebyshev models are prefixed with `chebmodel_`, and similarly `mpfi_chebpoly_` for polynomials with MPFI interval coefficients, `mpfr_chebpoly_` for polynomials with MPFR floating-point coefficients, etc. To briefly cover the implementation aspect, we provide after each following algorithm the corresponding C function name, together with implementation related remarks, when appropriate.

Coq formalization of RPAs

We briefly sketch out the formalization of RPAs in our Coq development. It is totally parametric with respect to the choice of basis, but only concerns *uniform approximations*, that is with the $\|\cdot\|_\infty$ norm.

A basis is described by a family of functions, non necessarily polynomials, indexed by natural numbers, that is a term `T: nat→R→R`. The structure `BasisOps_on` below describes the signature required on a basis `T`. It is parameterized by the type `C` of coefficients; sequences of such coefficients (`seq C`) represent linear combinations of elements of `T`. Linear operations (**Section 3.2.2**) need not be provided since they can be implemented independently from the basis. The range operation (**Section 3.2.1**) is important: its role is to bound the range on the given domain; it should be as accurate as possible since it is used at many places to compute error bounds in rigorous approximations (e.g., for multiplication and a posteriori validation). We define `BasisOps` to be a polymorphic function so that we capture with a single object the idealized operations on reals and their concrete implementation with intervals.

```
Record BasisOps_on (C: Type) := {
  lo, hi: C;                      (* bounds for the domain *)
  beval: seq C → C → C;           (* (efficient) evaluation *)
  bmul: seq C → seq C → seq C;    (* multiplication *)
  bone, bid: seq C;               (* constant to 1, identity *)
  bprim: seq C → seq C; }.        (* primitive *)
  brange: seq C → C*C; }.         (* range *)
Definition BasisOps := ∀ C: Ops1, BasisOps_on C.
```

Note that the type `C` used for the coefficients has a signature of an `Ops1`, which may represent any class of "numbers" with arithmetic operations, in particular intervals (cf. the complete article **[40,** Sec. 2**]** for more details about the underlying data-structures).

Then we can define RPAs (called *Models* in this framework) as follows:

```
Record Model C := { pol: seq C; rem: C }.
```

Again, `C` has to be thought as an abstract type for intervals. Therefore, this definition follows the polynomial – interval remainder approach rather than the polynomial – error bound one, since this Coq framework only deals with uniform approximations right now.

3.2 Elementary self-validating operations on rigorous polynomial approximations

3.2.1 ▶ Evaluation and range

An elementary requirement of rigorous polynomial approximations is to provide an interval extension for the function it represents, as defined in **Section 1.3**. Note however that not all norms are suitable for this requirement. The norm $\|\cdot\|_\infty$ is natural for this purpose, and $\|\cdot\|_{\mathrm{Q1}}$ is also compatible since its overapproximates $\|\cdot\|_\infty$. Other norms, like L^2 norms, are not compatible with this requirement. Instead of local evaluation, L^2 RPAs could be used for local integration.

In this section, we therefore suppose that $(\mathcal{F}, \|\cdot\|)$ is either $(\mathcal{C}^0(I), \|\cdot\|_\infty)$ or $(\mathrm{Y}^1, \|\cdot\|_{\mathrm{Y}^1})$. For a RPA $\mathbb{f} := (\mathbb{p}, \varepsilon)$ and an interval $\mathbb{t}$, the evaluation is simply defined as:

$$\textsc{RpaEval}(\mathbb{f}, \mathbb{t}) := \mathbb{p}(\mathbb{t}) \oplus [-1, 1] \otimes [\varepsilon], \tag{3.1}$$

which we may write $\mathbb{f}(\mathbb{t})$ for short. This depends on an evaluation scheme for polynomials (in the generalized sense) associated to the basis in which $\mathbb{p} = \sum_{i=0}^{n} \mathbb{a}_i \ell_i$ is considered. For the Chebyshev basis, we use the Clenshaw evaluation scheme (**Algorithm** Clenshaw in **Chapter 2**), overloaded with interval arithmetic operations.

Proposition 3.3 *For any RPA $\mathbb{f}$ and interval $\mathbb{t}$ contained in I, if $f \in \mathbb{f}$ and $t \in \mathbb{t}$, then $f(t) \in \mathbb{f}(\mathbb{t})$, implying that $\mathbb{f}(\cdot) : \mathbb{t} \mapsto \mathbb{f}(\mathbb{t})$ is an interval extension of f.*

Proof. Let $\mathbb{f} := (\mathbb{p}, \varepsilon)$. Since $f \in \mathbb{f}$, there exists $p \in \mathbb{p}$ and $e \in \mathcal{F}$ such that $f(t) = p(t) + e(t)$ and $\|e\| \leqslant \varepsilon$.

By hypothesis, $\mathbb{p}(\cdot) : \mathbb{t} \mapsto \mathbb{p}(\mathbb{t})$ is an interval extension over I for p. In particular, $p(t) \in \mathbb{p}(\mathbb{t})$. Moreover, $\|e\| \leqslant \varepsilon$ (with $\|\cdot\| = \|\cdot\|_\infty$ or $\|\cdot\|_{\mathrm{Y}^1}$) implies that $e(t) \in [-1, 1] \otimes [\varepsilon] = [-\varepsilon, \varepsilon]$.

This finally proves that $f(t) = p(t) + e(t) \in \mathbb{f}(\mathbb{t}) := \mathbb{p}(\mathbb{t}) \oplus [-1, 1] \otimes [\varepsilon]$. □

Algorithm RpaEval provides valid enclosure for any $\mathbb{t} \subseteq I$. However, this may not be an appropriate strategy for *large* intervals, due to the overapproximation phenomena of interval arithmetic showcased in **Section 1.3**. In particular, one cannot hope in general to get reasonably tight enclosures of the total range over I, which will be necessary for several operations in the following sections. For the Chebyshev basis, one can define the *range* over $[-1, 1]$ of a Chebyshev model $\mathbb{f} = (\mathbb{p}, \varepsilon)$ with $\mathbb{p} = \sum_{i=0}^{n} \mathbb{a}_i T_i$ as in **[129,** Chap. 4, Algo. 4.5.2**]**:

$$\begin{aligned} [\![\mathbb{f}]\!] &:= [\![p]\!] \oplus [-1, 1] \otimes [\varepsilon], \qquad \text{where} \\ [\![p]\!] &:= \mathbb{a}_0 \oplus [-1, 1] \otimes \mathbb{a}_1 \oplus \cdots \oplus [-1, 1] \otimes \mathbb{a}_n. \end{aligned}$$

Proposition 3.4 *Let $\mathbb{f} = (\mathbb{p}, \varepsilon)$ be a Chebyshev model. Then,*

(3.4 i) *for all $p \in \mathbb{p}$ and $t \in [-1, 1]$, $p(t) \in [\![p]\!]$;*

(3.4 ii) *for all $f \in \mathbb{f}$ and $t \in [-1, 1]$, $f(t) \in [\![f]\!]$.*

Proof. For (3.4 i), let $\mathbb{p} = \sum_{i=0}^{n} \mathbb{a}_i T_i$ and $p \in \mathbb{p}$, that is $p = \sum_{i=0}^{n} a_i T_i$ with $a_i \in \mathbb{a}_i$. Since $T_0(t) = 1$ and $T_i(t) \in [-1, 1]$ for all $i \geqslant 1$ and $t \in [-1, 1]$, $p(t) \in [\![\mathbb{p}]\!]$ follows from the correctness of interval arithmetic operations.

Now, (3.4 ii) follows from (3.4 i) by a similar argument to the proof of **Proposition 3.3**. □

Remark 3.5 *Although $\|\cdot\|_{\mathrm{Y}^1}$ is not dominated by $\|\cdot\|_\infty$ (i.e., there is no constant C such that $\|f\|_{\mathrm{Y}^1} \leqslant C\|f\|_\infty$ for all $f \in \mathrm{Y}^1$), the induced overapproximation is quite reasonable in practice.*

Moreover, we also define the *truncation* of a RPA to a given degree N, which is useful when working with a globally fixed degree. For $\mathbb{f} := (\mathbb{p}, \varepsilon)$ with $\mathbb{p} := \sum_{i=0}^{N'} \mathbb{a}_i \ell_i$ and $N' \geqslant N$:

$$\textsc{RpaTrunc}(\mathbb{f}, N) := (\sum_{i=0}^{N} \mathbb{a}_i \ell_i, \Delta(\varepsilon + [\![\sum_{i=N+1}^{N'} \mathbb{a}_i \ell_i]\!])). \tag{3.2}$$

Clearly, if $f \in \mathbb{f}$, then we still have $f \in \textsc{RpaTrunc}(\mathbb{f}, N)$.

Implementation in C ChebValid provides several routines to evaluate Chebyshev models on different input types, e.g., for machine double: `chebmodel_evaluate_d`, for rationals (mpq): `chebmodel_evaluate_q`, for mpfr: `chebmodel_evaluate_fr` or resp., for mpfi `chebmodel_evaluate_fi`.

Clenshaw evaluation scheme in Coq

The imperative-style for-loop of **Algorithm** Clenshaw in **Chapter 2** is translated into the following polymorphic tail-recursive function with two accumulators:

```
Fixpoint Clenshaw (C: Ops1) b c (p: seq C) x :=
  match p with
  | [] => c - x*b
  | a::q => Clenshaw c (a + 2*x*c - b) q x
  end.
Definition beval (C: Ops1) (p: seq C) x := Clenshaw 0 0 (rev p) x.
```

This code might look mysterious. It is justified by the following invariant on real numbers:

```
Lemma ClenshawR b c p x: Clenshaw b c p x = eval T (catrev p [c - 2*x*b; b]) x.
```

In the right-hand side, `catrev` is the function that reverses its first argument and catenate it with the second one. The proof is done by induction in just three lines, using the Coq tactic for ring equations.

3.2.2 ▶ Linear space operations

Natural extensions of linear space operations are easy to define on rigorous polynomial approximations. This moreover does not depend on the choice of norm and basis. For RPAs $\mathbb{f} = (\mathbb{p}, \varepsilon)$ and $\mathbb{g} = (\mathbb{q}, \eta)$, with polynomials $\mathbb{p} = \sum_{i=0}^{n} \mathbb{a}_i \ell_i$ and $\mathbb{q} = \sum_{i=0}^{m} \mathbb{b}_i \ell_i$, and an interval $\mathbb{λ}$, we define:

$$\mathbb{f} \boxplus \mathbb{g} := (\mathbb{p} \boxplus \mathbb{q}, \triangle(\varepsilon + \eta)) \quad \text{where} \quad \mathbb{p} \boxplus \mathbb{q} := \sum_{i=0}^{\max(n,m)} (\mathbb{a}_i \oplus \mathbb{b}_i) T_i,$$

$$\boxminus \mathbb{f} := (\boxminus \mathbb{p}, \varepsilon) \quad \text{where} \quad \boxminus \mathbb{p} := \sum_{i=0}^{n} (\ominus \mathbb{a}_i) T_i,$$

$$\mathbb{f} \boxminus \mathbb{g} := (\mathbb{p} \boxminus \mathbb{q}, \triangle(\varepsilon + \eta)) \quad \text{where} \quad \mathbb{p} \boxminus \mathbb{q} := \sum_{i=0}^{\max(n,m)} (\mathbb{a}_i \ominus \mathbb{b}_i) T_i,$$

$$\mathbb{λ} \boxdot \mathbb{f} := (\mathbb{λ} \boxdot \mathbb{p}, \triangle(\mathrm{mag}(\mathbb{λ})\varepsilon) \quad \text{where} \quad \mathbb{λ} \boxdot \mathbb{p} := \sum_{i=0}^{n} (\mathbb{λ} \otimes \mathbb{a}_i) T_i.$$

These operations are implemented in ChebValid under the self-evident names `chebmodel_add`, `chebmodel_neg`, `chebmodel_sub`, `chebmodel_scalar_mul_fi`.

3.2.3 ▶ Multiplication of RPAs

Besides linear operations on RPAs, we shall also define a multiplication operation $(\mathbb{f}, \mathbb{g}) \mapsto \mathbb{f} \boxtimes \mathbb{g}$. Here, we make the assumption that $(\mathcal{F}, \|\cdot\|)$ is a *Banach algebra* (see **Lemma 2.39**), that is, $\mathcal{F}$ is closed under multiplication and $\|fg\| \leqslant \|f\|\|g\|$. Note that the classical setting for uniform approximation $(\mathcal{C}^0(I), \|\cdot\|_\infty)$, where $\|\cdot\|_\infty$ is the supremum norm over a given compact segment I, is a Banach algebra. The $\mathcal{Y}^1$ space of absolutely convergent Chebyshev series is another example of Banach algebra.

Algorithm RpaMul$(\mathbb{f}, \mathbb{g}, N)$ computes a degree N RPA for the multiplication of two RPAs $\mathbb{f}$ and $\mathbb{g}$, provided a multiplication scheme $(p, q) \mapsto p \times q$ is given for the underlying basis.

Algorithm 3.1 RpaMul$(\mathbb{f}, \mathbb{g}, N)$ – Multiplication of RPAs

Input: RPAs $\mathbb{f} = (\mathbb{p}, \varepsilon)$ and $\mathbb{g} = (\mathbb{q}, \eta)$, degree N.
Output: A degree N RPA $\mathbb{h}$ for the multiplication of $\mathbb{f}$ and $\mathbb{g}$.

1: $\mathbb{r} \leftarrow \mathbb{p} \times \mathbb{q}$
2: $\tau \leftarrow \Delta(\llbracket \mathbb{p} \rrbracket \eta + \llbracket \mathbb{q} \rrbracket \varepsilon + \varepsilon\eta)$
3: $\mathbb{h} \leftarrow$ RpaTrunc$((\mathbb{r}, \tau), N)$
4: **return** $\mathbb{h}$

■ **Proposition 3.6** (Correctness of RpaMul) *If $f \in \mathbb{f}$ and $g \in \mathbb{g}$, then $fg \in$ RpaMul$(\mathbb{f}, \mathbb{g}, N)$, for any N.*

Proof. Let $\mathbb{f} = (\mathbb{p}, \varepsilon)$, $\mathbb{g} = (\mathbb{q}, \eta)$, $\mathbb{r}$ and τ be as in **Algorithm** RpaMul. Suppose that $f \in \mathbb{f}$ and $g \in \mathbb{g}$, meaning that there exist polynomials p and q such that:

$$p \in \mathbb{p}, \qquad q \in \mathbb{q}, \qquad \|f - p\| \leqslant \varepsilon, \qquad \|g - q\| \leqslant \eta.$$

Then, by the supposed correctness of polynomial multiplication with interval coefficients, $pq \in \mathbb{h} := \mathbb{p} \times \mathbb{q}$, and:

$$\|fg - pq\| = \|p(g-q) + q(f-p) + (f-p)(g-q)\| \leqslant \\ \|p\|\|g-q\| + \|q\|\|f-p\| + \|f-p\|\|g-q\| \leqslant \llbracket \mathbb{p} \rrbracket \eta + \llbracket \mathbb{q} \rrbracket \varepsilon + \varepsilon\eta \leqslant \tau$$

by the correctness of $\llbracket \cdot \rrbracket$ (**Proposition 3.4**).

Therefore, $fg \in (\mathbb{h}, \tau)$. Finally, thanks to the correctness of RpaTrunc, $fg \in \mathbb{h}$. □

Note that in the current version of ChebValid, the function `chebmodel_mul` does not truncate the result, which corresponds to $N = +\infty$ in RpaMul. However, future versions should give the possibility to fix a global degree N_{glob} for RPAs (similarly to a global floating-point precision fixed by the user), so that multiplication would be defined by:

$$\mathbb{f} \boxtimes \mathbb{g} := \text{RpaMul}(\mathbb{f}, \mathbb{g}, N_{\text{glob}}).$$

■ Multiplication in Chebyshev basis

In order to get a complete implementation of Chebyshev models, we must provide a multiplication scheme $(p, q) \mapsto p \times q$ for polynomials expressed in Chebyshev basis, in particular for Chebyshev polynomials with *interval* coefficients.

Following the discussion about fast multiplication in Chebyshev basis in the previous chapter, it might be tempting to instantiate fast DCT algorithms with intervals to obtain near-linear multiplication algorithms for the multiplication of Chebyshev models. Despite the rather good numerical conditioning of DCT with floating-point numbers [198], the interval counterpart is subject to ongoing works, as it may be subject to potentially large overestimations, similarly to what is observed in FFT [157].

In a recent work [45], the rounding errors occurring during a FFT are bounded *a priori*, thus opening the way to fast and rigorous polynomial multiplication in monomial basis. There is some hope for similar results for the DCT in the near future, which would lead to fast multiplication algorithms of Chebyshev models.

In the meantime, it is preferable to resort to traditional (quadratic-time) multiplication schemes. **Algorithm** ChebMul implements multiplication of Chebyshev polynomials as a straightforward application of the multiplication formula (2.6). It can be instantiated with rational numbers, floating-point numbers or intervals (the last case being used by **Algorithm** RpaMul for Chebyshev models). The ChebValid C library provides `mpfr_chebpoly_mul` and `mpfi_chebpoly_mul`.

Algorithm 3.2 ChebMul(p, q) – Multiplication in Chebyshev basis

Input: polynomials $p = \sum_{i=0}^{r} p_i T_i$ and $q = \sum_{j=0}^{s} q_j T_j$.

Output: polynomial $h = \sum_{k=0}^{r+s} h_k T_k$ representing the multiplication of p and q.

1: **for** $k = 0$ **to** $r + s$ **do** $h_k \leftarrow 0$
2: **for** $i = 0$ **to** r **do**
3: **for** $j = 0$ **to** s **do**
4: $h_{i+j} \leftarrow h_{i+j} + p_i * q_j/2$
5: $h_{|i-j|} \leftarrow h_{|i-j|} + p_i * q_j/2$
6: **end for**
7: **end for**
8: **return** h

Proposition 3.7 (Complexity of naive Chebyshev multiplication) *Algorithm* ChebMul(p, q) *computes the product of* $p := \sum_{i=0}^{r} a_i T_i$ *and* $q := \sum_{j=0}^{s} b_j T_j$ *in Chebyshev basis in* $\mathcal{O}(rs)$ *arithmetic operations.*

Therefore, the multiplication $\mathbb{f} \boxtimes \mathbb{g}$ *of Chebyshev models* $\mathbb{f}$ *and* $\mathbb{g}$ *with fixed degree* N_{glob} *and naive Chebyshev multiplication requires* $\mathcal{O}({N_{\text{glob}}}^2)$ *interval arithmetic operations.*

Proof. Clearly, the two nested for-loops in the code of ChebMul account for the claimed complexity in $\mathcal{O}(rs)$. Concerning $\mathbb{f} \boxtimes \mathbb{g}$, besides the multiplication in $\mathcal{O}({N_{\text{glob}}}^2)$ operations, the other routines involved in RpaMul (range $[\![\cdot]\!]$ and truncation RpaTrunc) have a linear complexity. □

A note concerning the Coq implementation of Chebyshev multiplication

Algorithm ChebMul can be implemented as a polymorphic function parametrized by the type of coefficients: `∀ (C : Ops1), seq C → seq C → seq C`. However, since lists in Coq do not allow for constant-time access to their elements, the resulting algorithm would run in cubic time.

To get around this issue, we propose two (quadratic-time) auxiliary functions `mul_pls` and `mul_mns`, implementing respectively the T_{i+j} and $T_{|i-j|}$ component of the multiplication formula. Then, `smul` is defined as the half sum of them:

```
Fixpoint mul_pls (C: Ops1) (p q: seq C): seq C :=
  match p,q with
  | [],_ | _,[] => []
  | a::p', b::q' =>
      sadd (a*b::(sadd (sscal a q') (sscal b p'))) (0::0::mul_pls p' q')
  end.

Fixpoint mul_mns (C: Ops1) (p q: seq C): seq C :=
  match p,q with
  | [],_ | _,[] => []
  | a::p', b::q' =>
      sadd (a*b::(sadd (sscal a q') (sscal b p'))) (mul_mns p' q')
  end.

Definition smul C (p q: seq C): seq C :=
  sscal (1/2) (sadd (mul_mns p q) (mul_pls p q)).
```

The multiplication of RPAs (in any basis, provided a multiplication scheme is given) is implemented by:

```
Definition mmul (M N: Model): Model :=
   {| pol := pol M * pol N;
      rem := srange (pol M) * rem N + srange (pol N) * rem M + rem M * rem N
   |}.
```

3.2.4 ▶ Integration

For the two examples of norms we have considered so far, that is, $\|\cdot\|_\infty$ (uniform convergence over a compact segment I) and $\|\cdot\|_{\mathrm{Q}^1}$, the (indefinite or definite) integration is a continuous linear operator. Hence, with these norms, integration of RPAs is possible:

- For the uniform convergence over $I = [a, b]$, the indefinite integral (a.k.a. primitive) taken from any $t_0 \in [a, b]$ is defined by:

$$\textsc{RpaPrim}_\infty((\mathbb{p}, \varepsilon), t_0) := \left(\int_{t_0} \mathbb{p}, \max(t_0 - a, b - t_0)\varepsilon \right). \tag{3.3}$$

  If a global degree N_{glob} is fixed, the resulting RPA (of degree $N_{\mathrm{glob}}+1$) must be truncated.

  The definite integral over $[c, d] \subseteq [a, b]$ returns the following interval:

$$\textsc{RpaInt}((\mathbb{p}, \varepsilon), c, d) := \left(\int_c \mathbb{p} \right)(d) \oplus [-1, 1] \otimes [(d - c)\varepsilon].$$

- Based on **Proposition 2.40**, the indefinite integration of RPAs from any $t_0 \in [-1,1]$ with the $\mathcal{Y}^1$ norm is:

$$\textsc{RpaPrim}_{\mathcal{Y}^1}((\mathbb{p},\varepsilon),t_0) := \left(\int_{t_0} \mathbb{p}, 2\varepsilon\right).$$

The integration scheme of the polynomials with interval coefficients depends on the choice of the basis. For the Chebyshev basis, **Algorithm** ChebInt computes a primitive using Formula (2.11).

Remark 3.8 (Derivation of RPAs) *The fact that $f_n \to f$ uniformly (resp. for the $\mathcal{Y}^1$-norm) does not imply that $f_n' \to f'$. In fact, f' may even not exist and belong to $\mathcal{C}^0(I)$ (resp. $\mathcal{Y}^1$). Hence, for the two norms considered in this section, differentiating RPAs is not possible.*

Algorithm 3.3 ChebInt(p, t_0) – Primitive in Chebyshev basis

Input: polynomial $p = \sum_{i=0}^{r} p_i T_i$ and $t_0 \in [-1,1]$.

Output: polynomial $q = \sum_{i=0}^{r+1} q_i T_i$ representing $\int_{t_0}^{t} p(s)\mathrm{d}s$.

```
for i = 0 to r + 1 do q_i ← 0
for i = 0 to r do
    if i = 0 then
        q_1 ← q_1 + p_0
    else if i = 1 then
        q_2 ← q_2 + p_1/4
    else
        q_{i+1} ← q_{i+1} + p_i/2(i + 1)
        q_{i-1} ← q_{i-1} − p_i/2(i − 1)
    end if
end for
q_0 ← q_0 − q(t_0)
return q
```

3.3 A posteriori fixed-point based validation methods

Let X denote an ambient space in which we look for the solution x^* of a given problem. In this setting, X is usually (at least) a complete metric space with a metric d. Depending on the problem under consideration, there may be no obvious *self-validating* algorithm to compute certified representations of x^*. Several obstructions may appear. For instance, **Example 1.30**

illustrated that merely replacing floating-point numbers by intervals in a standard Gaussian elimination is rarely a good option for rigorous matrix inversion – except for low dimension – because of rapidly increasing interval width, even for well-conditioned matrices. What is more, using intervals in approximation schemes does not even produce rigorous enclosures, because it cannot account for *method errors*. Consider, for example, polynomial interpolation or Runge-Kutta integration of ODEs.

In such cases, a possible approach consists in bounding separately rounding and method errors, and return their sum. Taking back the example of Runge-Kutta schemes, estimates for the method errors exist in the literature (see for example **[124,** Chap. 3**]**), and a priori rounding error bounds are automatically computed in **[28]**. However, this general process rapidly becomes cumbersome. Indeed, reasonably tight rounding error bounds may be hard to compute, whereas the method error estimates are often only asymptotically valid or may depend on unknown quantities (e.g., higher order derivatives).

A completely different approach concerns fixed-point based a posteriori validation methods. Dating back to the works of Kantorovich about Newton's method **[131]**, they gained prominence with the rise of modern computers and were applied to numerous functional analysis problems. Early works include **[136, 137]**, in which the use of fixed-point theorems is advocated to numerically prove the existence of solutions in a certain region. Based on those ideas, a wide range of works proposed methods to rigorously solve various differential problems, such as ODEs with initial or boundary conditions **[199, 189, 153]**, or elliptic PDEs **[200, 178, 9]**. The interested reader will find more details and references in the detailed survey **[179]** or in the short notice **[251]**. Even more recently, those methods were used to design algorithms computing RPAs for solutions of linear ODEs (see **[19]** and **Chapter 4**). In short, these methods compute a rigorous representation $\mathbb{x}$ for x^* in two separate steps:

1. **Approximation step.** A numerical approximation $x^\circ \in X$ of x^* is obtained by an oracle, which may resort to any approximation method and requires no mathematical assumption (convergence, error bounds...). This makes it possible to rely on external code and use (non-rigorous) libraries known for their highly efficient and optimized routines (e.g., matrix inversion with BLAS/LAPACK **[4]** or Chebyshev approximations with Chebfun **[72]**). In the perspective of rigorous numerics formalized inside computer assistants, this is a major breakthrough: time-consuming computations are independently executed, and the trusted codebase remains small.

2. **Validation step.** The initial problem is rephrased in such a way that x^* is a fixed point of a (locally) contracting operator $\mathbf{T} : X \to X$. An a posteriori error bound ε on $d(x^\circ, x^*)$ is deduced from the Banach fixed-point theorem (several variants of this theorem are given in **Section 3.3.1**). Obtaining an appropriate fixed-point equation for x^* is an essential part. To this aim, Newton-like methods have been widely used in the rigorous numerics literature **[265, 153]**.

3.3.1 ▶ The Banach fixed-point theorem and a posteriori Newton-like validation methods

A wide range of a posteriori validation methods rely on topological fixed-point theorems, the most classical ones in this context being the *Banach fixed-point theorem* and the *Schauder fixed-point theorem*. In this thesis, we focus on validation methods using the former. The interested reader will find examples involving the latter in, e.g., **[136,** Chap. 2, Thm. 4**]**, **[199]** or **[181]**.

After providing a statement of the Banach fixed-point theorem suitable for effective validation, we present the general ideas of *a posteriori Newton-like validation methods* that allow for turning general equations in Banach spaces into desired fixed-point equations.

Several statements of Banach fixed-point theorem

Theorem 3.9 is the traditional formulation of the Banach fixed-point theorem. It is a global statement – the operator $\mathbf{T}$ is required to be contracting over the whole space X. The classical proof that may be found in numerous textbooks (see for example **[20,** Thm. 2.1**]**) involves Cauchy sequences. We reproduce it here for the sake of completeness. It is interesting to note that our formalized proof based on filters and given in **Section 3.3.2** is somewhat different.

Theorem 3.9 (Banach fixed-point – global version) *Let (X, d) be a complete metric space and $\mathbf{T} : X \to X$ be an operator. If $\mathbf{T}$ is contracting over X, that is, if there exists a $\mu \in [0, 1)$ such that $\mathbf{T}$ is μ-Lipschitz over X:*

$$\forall x_1, x_2 \in X, \quad d(\mathbf{T} \cdot x_1, \mathbf{T} \cdot x_2) \leqslant \mu d(x_1, x_2),$$

then $\mathbf{T}$ admits a unique fixed point x^* in X.

Moreover, for any $x^\circ \in X$, the approximation error of x° to x^* satisfies the following inequality:

$$\frac{d(x^\circ, \mathbf{T} \cdot x^\circ)}{1 + \mu} \quad \leqslant \quad d(x^\circ, x^*) \quad \leqslant \quad \frac{d(x^\circ, \mathbf{T} \cdot x^\circ)}{1 - \mu}. \tag{3.4}$$

Proof. Let $x_n := \mathbf{T}^n \cdot x^\circ$ for $n \geqslant 0$ denote the iterates of x° under $\mathbf{T}$, that is $x_0 := x^\circ$ and $x_{n+1} := \mathbf{T} \cdot x_n$. Let moreover $b := d(x^\circ, \mathbf{T} \cdot x^\circ)$.

By an easy induction, one gets $d(x_n, x_{n+1}) \leqslant \mu^n b$. Since $\mu < 1$, this yields for all n and $m \geqslant n$:

$$d(x_n, x_m) \leqslant \sum_{k=n}^{m-1} \mu^k b \leqslant \sum_{k=n}^{+\infty} \mu^k b = \frac{\mu^n b}{1 - \mu} \to 0 \quad \text{as } n \to +\infty.$$

Therefore, (x_n) is a Cauchy sequence, and hence converges to some $x^* \in X$ by completeness. Moreover, $\mathbf{T} \cdot x^* = x^*$ since $\mathbf{T}$ is Lipschitz, hence continuous.

Now, to prove the uniqueness, let $\bar{x}, x^* \in X$ be two fixed points of $\mathbf{T}$. Then we have:

$$d(\bar{x}, x^*) = d(\mathbf{T} \cdot \bar{x}, \mathbf{T} \cdot x^*) \leqslant \mu d(\bar{x}, x^*),$$

from which follows $d(\bar{x}, x^*) = 0$, hence $\bar{x} = x^*$ since $\mu < 1$.

Finally, the enclosure (3.4) is proved by using the triangle inequality:

$$\begin{aligned} d(x^\circ, x^*) &\leqslant d(x^\circ, \mathbf{T} \cdot x^\circ) + d(\mathbf{T} \cdot x^\circ, x^*) \leqslant d(x^\circ, \mathbf{T} \cdot x^\circ) + \mu d(x^\circ, x^*), \\ d(x^\circ, \mathbf{T} \cdot x^\circ) &\leqslant d(x^\circ, x^*) + d(x^*, \mathbf{T} \cdot x^\circ) \leqslant d(x^\circ, x^*) + \mu d(x^\circ, x^*). \end{aligned}$$

□

It is quite often the case that the operator $\mathbf{T}$ is contracting only over some *neighborhood* of the candidate approximation x°. In fact, one can restrict the ambient metric space X to any closed stable subset $S \subseteq X$, since S (with the induced metric d) is again a complete metric space. However, for the purpose of *effective* validation, we give the following *local* statement. By replacing the general – and difficult to check – notion of closed stable subset with that of *strongly stable ball* (cf. the more detailed **Definition 3.14**), we obtain a formulation where rigorously verifying the preconditions only requires bounding distances and checking real inequalities.

Theorem 3.10 (Banach fixed-point – local version) *Let (X, d) be a complete metric space, an operator $\mathbf{T} : X \to X$, $x^\circ \in X$, and $\mu, b, r \in \mathbb{R}_+$, satisfying the following conditions:*

(3.10 i) $d(x^\circ, \mathbf{T} \cdot x^\circ) \leqslant b$;

(3.10 ii) $\mathbf{T}$ *is μ-Lipschitz over the closed ball* $\bar{B}(x^\circ, r) := \{x \in X \mid d(x, x^\circ) \leqslant r\}$:

$$\forall x_1, x_2 \in X, \quad x_1 \in \bar{B}(x^\circ, r) \ \wedge \ x_2 \in \bar{B}(x^\circ, r) \ \Rightarrow \ d(\mathbf{T} \cdot x_1, \mathbf{T} \cdot x_2) \leqslant \mu d(x_1, x_2);$$

(3.10 iii) $\mu < 1$ — $\mathbf{T}$ *is contracting over* $\bar{B}(x^\circ, r)$;

(3.10 iv) $b + \mu r \leqslant r$ — $\bar{B}(x^\circ, r)$ *is called a* μ-strongly stable ball with offset b.

Then $\mathbf{T}$ admits a unique fixed-point x^* in $\bar{B}(x^\circ, r)$.

Proof. The ball $\bar{B}(x^\circ, r)$ is stable under $\mathbf{T}$. Indeed, for any $x \in \bar{B}(x^\circ, r)$,

$$d(x^\circ, \mathbf{T} \cdot x) \leqslant d(x^\circ, \mathbf{T} \cdot x^\circ) + d(\mathbf{T} \cdot x^\circ, \mathbf{T} \cdot x) \leqslant b + \mu r \leqslant r.$$

Since $\mathbf{T}$ is contracting over $\bar{B}(x^\circ, r)$, **Theorem 3.9** – applied to the closed subspace $\bar{B}(x^\circ, r)$ instead of X – guarantees the existence and uniqueness of a fixed point x^* of $\mathbf{T}$ in $\bar{B}(x^\circ, r)$. □

Remark 3.11 *The reader may wonder why we do not provide an enclosure of $d(x^\circ, x^*)$ in **Theorem 3.10**, similarly to (3.4). In fact, the lower bound is not used in practice – it just tells that the enclosure is sharp for small μ – and **Condition (3.10 iv)** is equivalent to $r \geqslant \frac{b}{1-\mu}$, so that one can directly choose $r := \frac{b}{1-\mu}$ without violating the four conditions of the theorem.*

A posteriori Newton-like validation method

In order to use the Banach fixed-point theorem for validating a candidate approximation x° with respect to the exact solution x^*, one needs to find a contracting operator $\mathbf{T}$ of which x^* is a fixed point. To this end, one can resort to Newton-like validation methods, which transform an equation $\mathbf{F} \cdot x = 0$ into an equivalent fixed-point equation $\mathbf{T} \cdot x = x$ with $\mathbf{T}$ contracting **[153]**.

More specifically, suppose that $\mathbf{F} : X \to Y$ is differentiable; we use a Newton-like operator $\mathbf{T} : X \to X$ defined as:

$$\mathbf{T} \cdot x := x - \mathbf{A} \cdot \mathbf{F} \cdot x, \qquad \text{for } h \in X,$$

with $\mathbf{A} : Y \to X$ an *injective bounded linear* operator, intended to be close to $(\mathcal{D}\mathbf{F}_{x^\circ})^{-1}$. The operator $\mathbf{A}$ may be given by an oracle and does not need to be this exact inverse (which anyway might not be exactly representable on computers). The mean value theorem yields a Lipschitz ratio μ for $\mathbf{T}$ over any convex subset S of X:

$$\forall x_1, x_2 \in S, \ \|\mathbf{T} \cdot x_1 - \mathbf{T} \cdot x_2\| \leqslant \mu \|x_1 - x_2\|, \quad \text{with } \mu = \sup_{x \in S} \|\mathcal{D}\mathbf{T}_x\| = \sup_{x \in S} \|\mathbf{1}_X - \mathbf{A} \cdot \mathcal{D}\mathbf{F}_x\|,$$

which is expected to be small over some neighborhood of x°.

Concretely, in order to apply **Theorem 3.10**, one needs to compute the following quantities:

- a bound $b \geqslant \|\mathbf{A} \cdot \mathbf{F} \cdot x^\circ\| = \|x^\circ - \mathbf{T} \cdot x^\circ\|$;
- a bound $\mu_0 \geqslant \|\mathbf{1}_X - \mathbf{A} \cdot \mathcal{D}\mathbf{F}_{x^\circ}\| = \|\mathcal{D}\mathbf{T}_{x^\circ}\|$;
- a bound $\bar{\mu}(r) \geqslant \|\mathbf{A} \cdot (\mathcal{D}\mathbf{F}_x - \mathcal{D}\mathbf{F}_{x^\circ})\| = \|\mathcal{D}\mathbf{T}_x - \mathcal{D}\mathbf{T}_{x^\circ}\|$ valid for any $x \in \bar{B}(x^\circ, r)$, and parameterized by a radius $r \in \mathbb{R}_+$.

If we are able to find a radius $r \in \mathbb{R}_+$ satisfying:

$$\mu(r) := \mu_0 + \bar{\mu}(r) < 1, \qquad \text{and} \qquad b + r\mu(r) \leqslant r, \tag{3.5}$$

then **Theorem 3.10** guarantees the existence and uniqueness of a root x^* of $\mathbf{F}$ in $\bar{B}(x^\circ, r)$.

Remark 3.12 *Finding an r as small as possible while satisfying (3.5) may be a nontrivial task for automated validation procedures. For many problems, $\bar{\mu}(r)$ is polynomial, hence conditions (3.5) are polynomial inequalities over r (see, e.g., [199], or the more recent work [111], in which this approach is given the name* radii polynomials*). In our case, division (resp. square root) induces an affine (resp. quadratic) equation, which admits closed form solutions.*

Example 3.13 *To illustrate this Newton-like validation procedure, in contrast with the interval Newton method presented in **Section 1.3.3**, we consider the golden ratio of **Example 1.29**. Here, the ambient Banach space is the standard real line $(\mathbb{R}, |\cdot|)$, and $\varphi^* = \frac{1+\sqrt{5}}{2}$ is the unique positive root of the quadratic operator $\mathbf{F} : x \mapsto x^2 - x - 1$. We propose to validate the approximation $\varphi^\circ = \frac{55}{34}$.*

Following the above approach, since $\mathcal{D}\mathbf{F}_x \cdot h = (2x-1)h$, we define $\mathbf{A} = (\mathcal{D}\mathbf{F}_{\varphi^\circ})^{-1}$ as the multiplication by $a := \frac{1}{2\varphi^\circ - 1}$, yielding the following Newton-like operator $\mathbf{T}$:

$$\mathbf{T} \cdot x = x - a(x^2 - x - 1), \qquad \text{where} \quad a := \frac{1}{2\varphi^\circ - 1} = \frac{17}{38}.$$

Note that in this particular example, since we are working in $\mathbb{R}$ with exact rational numbers, $\mathbf{A}$ is the exact inverse of $\mathcal{D}\mathbf{F}_{\varphi^\circ}$, implying $\mu_0 = 0$ in the above framework.

- *First, consider a ball $\bar{B}(\varphi^\circ, r)$ of some radius r around φ°. A Lipschitz ratio for $\mathbf{T}$ over it can be explicitly computed. Indeed,*

$$\begin{aligned}\mathbf{T} \cdot x_1 - \mathbf{T} \cdot x_2 &= (x_1 - x_2) - \frac{(x_1^2 - x_2^2) - (x_1 - x_2)}{2\varphi^\circ - 1} \\ &= \left(1 - \frac{(x_1 + x_2) - 1}{2\varphi^\circ - 1}\right)(x_1 - x_2) = \left(-\frac{(x_1 - \varphi^\circ) + (x_2 - \varphi^\circ)}{2\varphi^\circ - 1}\right)(x_1 - x_2).\end{aligned}$$

  yields a valid Lipschitz ratio $\mu(r) := 2ar$.

- *We also compute the bound:*

$$b := |\varphi^\circ - \mathbf{T} \cdot \varphi^\circ| = |a({\varphi^\circ}^2 - \varphi^\circ - 1)| = \frac{1}{2584}.$$

- *Now, $\bar{B}(\varphi^\circ, r)$ contains a unique fixed point of $\mathbf{T}$ as soon as r satisfies the inequalities:*

$$\mu(r) := \frac{17}{19} r < 1, \qquad \textit{and} \qquad b + r\mu(r) := \frac{1}{2584} + \frac{17}{19} r^2 \leqslant r.$$

Therefore, we can for example provide the a posteriori error bound $r := \frac{39}{10000}$. Since φ^* is the unique positive root of $\mathbf{F}$ and that $\bar{B}(\varphi^\circ, r)$ clearly contains only positive numbers, we claim the following rigorous statement:

$$|\varphi^\circ - \varphi^*| \leqslant \frac{39}{100000}.$$

In comparison, MATHEMATICA *numerically computes $|\varphi^\circ - \varphi^*| \approx 0.00038693$.*

3.3.2 ▶ A formal proof for the Banach fixed-point theorem based on filters

The Banach fixed-point theorem has been formalized in various flavors of logic and proof assistants. In particular, a formal proof of a version of this fixed-point theorem is provided in [27] for the purpose of the formalization of the Lax-Milgram theorem, based on the COQUELICOT library [29]. Using the same backbone library, we provide a different formalization, in accordance with the statement of **Theorem 3.10**, that is more suitable for our effective validation purposes.

The COQUELICOT library for functional analysis

The COQUELICOT library [29] formalizes topological concepts using *filters* [33, 108], which we briefly recall here. A filter on a type T is a collection of collections of inhabitants of T which is non-empty, upward closed and stable under finite intersections:

```
Record Filter (T : Type) (F : (T → Prop) → Prop) := {
  filter_true : F (fun _ => True) ;
  filter_and : ∀ P Q : T → Prop, F P → F Q → F (fun x => P x /\ Q x) ;
  filter_imp : ∀ P Q : T → Prop, (∀ x, P x → Q x) → F P → F Q }.
```

While filters are used to formalize neighborhoods, *balls* allow for expressing the relative closeness of points in the space. Balls are formalized using a ternary relation between two points in the carrier type, and a real number, with the following axioms:

```
  ball : M → R → M → Prop ;
  ax1 : ∀ x (e > 0), ball x e x ;
  ax2 : ∀ x y e, ball x e y → ball y e x ;
  ax3 : ∀ x y z e1 e2, ball x e1 y → ball y e2 z → ball x (e1 + e2) z
```

Two points are called *close* when they cannot be separated by balls:

```
Definition close (x y : M) : Prop := ∀ eps > 0, ball x eps y.
```

A filter is called a *Cauchy filter* when it contains balls of arbitrary (small) radius:

```
Definition cauchy (T : UniformSpace) (F : (T → Prop) → Prop) :=
  ∀ eps > 0, ∃ x, F (ball x eps).
```

Finally, a *uniform space* is a type equipped with a ball relation and a *complete space* moreover has a limit operation on filters, which ensures the convergence of Cauchy sequences via the following axioms (where `ProperFilter F` is equivalent to `Filter F /\ ∀ P, F P → ∃ x, P x`):

```
lim : ((T → Prop) → Prop) → T ;
ax1 : ∀ F, ProperFilter F → cauchy F → ∀ eps > 0, F (ball (lim F) eps) ;
ax2 : ∀ F1 F2, F1 ⊆  F2 → F2 ⊆ F1 → close (lim F1) (lim F2)
```

The above formal definition of balls does not enforce closedness nor openness. We thus introduced the relation associated with the *closure* of balls, so as to model *closed* neighborhoods:

```
Definition cball x r y := ∀ e > 0, ball x (r+e) y.
```

Equipped with this definition, **Hypothesis (3.10 ii)** of **Theorem 3.10** is formalized as follows:

```
Definition lipschitz_on (F : U → U) (mu : R) (P : U → Prop) :=
∀ x y : U, ∀ r ⩾ 0, P x → P y → cball x r y → cball (F x) (mu*r) (F y).
```

Formal proof of Banach fixed-point theorem

We now sketch the formalized proof, using mathematical notations for the sake of simplicity. We consider a complete space X and we write $y \in B(x,r)$ for the formal `(ball x r y)`, and $y \in \bar{B}(x,r)$ for `(cball x r y)`. The key notion is that of *strongly stable ball*:

Definition 3.14 (Strongly stable ball) *A ball $\bar{B}(u,r)$ is* μ-strongly stable *for* $\mathbf{T}$ *if* $\mathbf{T}$ *is* μ*-Lipschitz on* $\bar{B}(u,r)$ *and if there is a nonnegative real number* s*, called the* offset*, s.t.:*

$$\mathbf{T}\cdot u \in \bar{B}(u,s) \qquad \textit{and} \qquad s+\mu r \leqslant r.$$

Remark 3.15 (Stability) *For any x in $\bar{B}(u,r)$, a strongly stable ball for $\mathbf{T}$, $\mathbf{T}\cdot x \in \bar{B}(u,r)$.*

Remark 3.16 (Contracting case) *When $0 \leqslant \mu < 1$, for any μ-strongly stable ball $\bar{B}(v,\rho)$, with offset σ, $\bar{B}(\mathbf{T}\cdot v, \mu\rho)$ is also a strongly stable ball, with offset $\mu\sigma$. Moreover, $\bar{B}(\mathbf{T}\cdot v,\mu\rho)$ is included in $\bar{B}(v,\rho)$.*

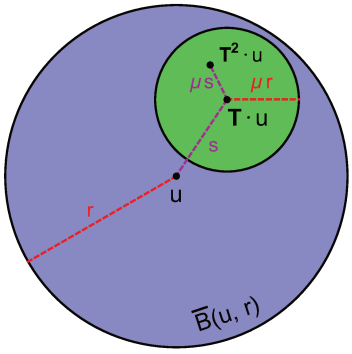


Figure 3.1: Balls $\overline{B}_0$ and $\overline{B}_1$

Assume that $\mathbf{T}$ has a μ-strongly stable ball $\bar{B}(u,r)$ of offset s, with $\mu < 1$. In particular, $\mathbf{T}$ is contracting on $\bar{B}(u,r)$. Consider the sequence of balls defined as follows:

$$\bar{B}_n = \bar{B}(u_n, r_n) \quad \text{with} \quad u_n = \mathbf{T}^n \cdot u \quad \text{and} \quad r_n = r\mu^n$$

where $\mathbf{T}^n \cdot u$ denotes the iterated images of u under $\mathbf{T}$. By **Remark 3.16**, $(\bar{B}_n)_{n\in\mathbb{N}}$ is a nested sequence of μ-strongly stable ball for $\mathbf{T}$, with offset $s\mu^n$. Let $\mathcal{F}$ be the family of collections of points in U defined as:

$$\mathcal{F} := \{P \subseteq U \mid \exists\, n, \bar{B}_n \subseteq P\}.$$

It is a proper filter: $\mathcal{F}$ contains U, it is obviously upward closed, and for $P, Q \in \mathcal{F}$, $P \cap Q$ is also in $\mathcal{F}$ because $(\bar{B}_n)_{n\in\mathbb{N}}$ is decreasing for inclusion. Thus $\mathcal{F}$ has a limit w, such that for any $\varepsilon > 0$, balls $\bar{B}_n$ are eventually included in $B(w,\varepsilon)$. We provide a formal proof of **Theorem 3.17**, a reformulation of **Theorem 3.10** using the vocabulary of the Coquelicot library:

Theorem 3.17 *The limit w of the filter $\mathcal{F}$ is in $\bar{B}_0$, and w is a fixed point of $\mathbf{T}$. Moreover, w is close to every other fixed point of $\mathbf{T}$ in $\overline{B}_0$.*

Proof. In this statement "w is a fixed point of $\mathbf{T}$" means "w is close to $\mathbf{T} \cdot w$". First, $w \in \bar{B}_n$ for all n. Indeed, for any $\varepsilon > 0$, there is an $m \geqslant n$ s.t. $\bar{B}_m \subseteq B(w, \varepsilon)$, and since $\bar{B}_m \subseteq \bar{B}_n$, $u_m \in \bar{B}_n \cap B(w, \varepsilon)$. In particular, $w \in \bar{B}_0$.

It is also clear by stability that $\mathbf{T} \cdot w \in \bar{B}_n$ for all n.

Moreover, w is close to any point v s.t. $v \in \bar{B}_n$ for all n (for any $\varepsilon > 0$, choose n s.t. $2\mu r^n < \varepsilon$). Taking $v := \mathbf{T} \cdot w$ proves that w is a fixed point of $\mathbf{T}$.

Finally, if $w' \in \bar{B}_0$ is another fixed point of $\mathbf{T}$, then it follows from an easy induction that $w' \in \bar{B}_n$ for all n. Hence, the foregoing shows that w is close to w'. □

Strongly stable balls model the requirements set on the untrusted data to be formally verified. They can also be seen as balls centered at the initial point, and large enough to include all its successive iterates, i.e. as instances of the locus at stake in classical presentations of the proof. The version proved in **[27]** has a slightly more technical wording, which seems to be made necessary by its further usage in the verification of the Lax-Milgram theorem. Our proof script is significantly shorter, partly because we automate proofs of positivity conditions (for radii of balls) using canonical structures for manifestly positive expressions. But the key ingredient for concision is to make most of the filter device in the proof, and to refrain from resorting to low-level properties of geometric sequences. To the best of our knowledge, the other libraries of formalized analysis featuring a proof of this result, notably Isabelle/HOL and HOL-Light, are based on variants of proof strategy closer to the approach of Boldo et al. than to ours.

3.3.3 ▶ Application to the rigorous inversion of matrices

As illustrated by **Example 1.30** in **Chapter 1**, inverting a square matrix with a standard Gaussian elimination procedure overloaded with interval operations rarely offers a realistic solution, due to excessive interval overapproximations. The Newton-like a posteriori validation framework presented above yields a rather elementary method to compute tight error bounds for numerical matrix inverses. Although it is really close to the well-known Krawczyk method in the linear case, we give it here for the mere sake of illustration. An extensive literature is dedicated to this problem, and we refer the interested reader to **[174,** Chap. 7**]** and **[221,** Sec. 10**]** for more details on these methods.

Fix $n \geqslant 1$ and consider the set $\mathcal{M}_n(\mathbb{R})$ of order n square matrices with real entries. To obtain a Banach space, one needs a suitable norm on it. For the sake of simplicity, we restrict ourselves to the 1-norm $\|\cdot\|_1$, since it is the operator norm associated to the 1-norm in $\mathbb{R}$: $\|x\|_1 := |x_1| + \cdots + |x_n|$, whence a strong connection with the space Υ^1 of absolutely summable Chebyshev series (**Definition 2.37** in previous chapter).

Definition 3.18 *For $n \geqslant 1$, the 1-norm $\|\cdot\|_1$ over $\mathcal{M}_n(\mathbb{R})$, defined as*

$$\|M\|_1 := \max_{1 \leqslant j \leqslant n} \sum_{i=1}^{n} |m_{ij}|, \qquad M = (m_{ij})_{1 \leqslant i,j \leqslant n} \in \mathcal{M}_n(\mathbb{R}),$$

makes $(\mathcal{M}_n(\mathbb{R}), \|\cdot\|_1)$ a Banach algebra, that is:

$$\|MN\|_1 \leqslant \|M\|_1 \|N\|_1, \qquad M, N \in \mathcal{M}_n(\mathbb{R}).$$

Let P be an order n square matrix, expected to be invertible – but this remains to be proved – and M° a numerically computed inverse. If P is invertible, then $M^* = P^{-1}$ is the unique root of $\mathbf{F} : \mathcal{M}_n(\mathbb{R}) \to \mathcal{M}_n(\mathbb{R})$ defined as $\mathbf{F} \cdot M := PM - 1_n$, where 1_n is the order n identity matrix. This operator is affine, of linear part $M \mapsto PM$. Hence $\mathbf{A}$ can be chosen as $M \mapsto AM$ with some $A \approx P^{-1}$. We take $A := M^\circ$.

Based on this choice, Algorithm ValidatedMatrixInverse(P) computes, if possible, a pair (M°, ε) such that $\|M^\circ - P^{-1}\|_1 \leqslant \varepsilon$. In particular, this means that all the entries of M° have an error at most ε, allowing to construct an interval matrix $\mathbb{M}$ such that $P^{-1} \in \mathbb{M}$.

Algorithm 3.4 ValidatedMatrixInverse(P)

Input: an order n square matrix $P \in \mathcal{M}_n(\mathbb{R})$.
Output: a pair $(M^\circ, \varepsilon) \in \mathcal{M}_n(\mathbb{R}) \times \mathbb{R}$ such that $\|M^\circ - P^{-1}\|_1 \leqslant 1$, or "Fail".

▷ *Approximation step*
Compute M° as a floating-point numerical inverse of P

▷ *Validation step – use interval arithmetic*
$\mathbb{u} \leftarrow \|1_n - M^\circ P\|_1$
$\mathbb{b} \leftarrow \|M^\circ(PM^\circ - 1_n)\|_1$
if $\mathbb{u} < 1$ **then**
  $\mathbb{r} \leftarrow \mathbb{b} \oslash (\mathbb{1} \ominus \mathbb{u})$
  return $\varepsilon := \overline{r}$
else
  return "Fail"
end if

Proposition 3.19 (Correctness of ValidatedMatrixInverse) *For $P \in \mathcal{M}_n(\mathbb{R})$, if* ValidatedMatrixInverse(P) *returns a pair (M°, ε), then P is invertible and $\|M^\circ - P^{-1}\|_1 \leqslant \varepsilon$.*

Proof. Suppose that ValidatedMatrixInverse(P) returns such a pair (M°, ε). First, we have that $\mu := \|1_n - M^\circ P\|_1 \in \mathbb{u}$, hence $\mu < 1$. But if P is not invertible, then there exists $x \in \mathbb{R}^n$ of 1-norm $\|x\|_1 = 1$ in the kernel of P, yielding $\|(1_n - M^\circ P)x\|_1 = \|x\|_1 = 1$, which contradicts $\mu < 1$ (since the 1-norm of matrices is the operator norm corresponding to the 1-norm in $\mathbb{R}^n$). Therefore P is invertible.

Now we consider the Newton-like operator $\mathbf{T} : \mathcal{M}_n(\mathbb{R}) \to \mathcal{M}_n(\mathbb{R})$, defined by:

$$\mathbf{T} \cdot M := M - M^\circ(PM - 1_n).$$

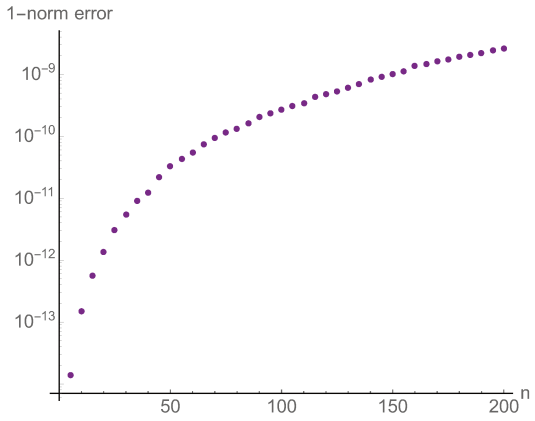
This operator is contracting over $\mathcal{M}_n(\mathbb{R})$, since

$$\|\mathbf{T} \cdot M_1 - \mathbf{T} \cdot M_2\| = \|(1_n - M^\circ P)(M_1 - M_2)\|_1 \leqslant \|1_n - M^\circ P\|_1 \|M_1 - M_2\|_1 = \mu \|M_1 - M_2\|_1.$$

Moreover, $b := \|M^\circ - \mathbf{T} \cdot M^\circ\|_1 = \|M^\circ(PM^\circ - 1_n)\|_1 \in \mathbb{b}$.

Thanks to the Banach fixed-point **Theorem 3.9**, we have that $\|M^\circ - P^{-1}\|_1 \leqslant \frac{b}{1-\mu} \in \mathbb{r}$. □

Example 3.20 (A posteriori validation at the rescue for Lehmer matrix) *Overloading floating-point operations by interval ones in Gaussian elimination is a classical example illustrating the limits of interval arithmetic, as shown by* ***Example 1.30*** *with Lehmer matrices.*



(a) Certified $\|\cdot\|_1$ error bound.

n	time (ms)	
	approx.	valid.
20	0	6
40	0	37
60	1	111
80	2	260
100	4	504
120	8	896
140	13	1506
160	21	2910
180	32	4455
200	49	6588

(b) Approximation and validation timings for L_n.

■ **Figure 3.2**: Rigorous inversion of Lehmer matrices L_n using ValidatedMatrixInverse.

Now, using ***Algorithm*** ValidatedMatrixInverse*, we obtain reasonably tight rigorous inverses.*

Figure 3.2a *plots the evolution of the computed* $\|\cdot\|_1$ *error in function of* n *for Lehmer matrices* L_n*, using standard double-precision for underlying floating-point arithmetic. Although the validation step is only quadratic, compared to the cubic complexity of Gaussian elimination in the approximation step, the timings given in* ***Table 3.2b*** *clearly put into evidence the efficiency gap between hardware double-precision arithmetic and* MPFI *multiprecision interval arithmetic. Note that, in this example, standard double precision is sufficient, so that we can reduce the computation time of the validation step by using intervals with machine floating-point endpoints, instead of* MPFI*.*

3.4 Division and square root of rigorous polynomial approximations using a posteriori validation

Division and square root of RPAs cannot be computed by straightforward self-validating algebraic methods, since the ring of polynomials is not closed under these operations. A standard approach used in **[164,** Sec. 4.3.2**]** for Taylor models, and in **[129,** Secs. 2.2.2 and 4.5.5**]** for Taylor and Chebyshev models, relies on composition. Specifically, given an RPA $\mathbb{f}$ for the operand and another one $\mathbb{g}$ for the function $x \mapsto 1/x$ (resp. $x \mapsto \sqrt{x}$) over an interval covering the range of $\mathbb{f}$, composing $\mathbb{g}$ with $\mathbb{f}$ allows for constructing a RPA for the reciprocal (resp. the square root) of $\mathbb{f}$. Since composition is realized using a linear number of operations on RPAs **[129,** Sec. 4.5.5**]**, that is, *cubic* complexity in interval operations, so are the composition-based division and square root.

Making use of the fundamentals of Banach fixed-point based a posteriori validation and Newton-like methods explained in the previous section, we propose rigorous division and square root procedures for RPAs with *quadratic* complexity (**Propositions 3.23** and **3.25**). For both operations, the roadmap is the following. First, we give an "*abstract*" validation procedure under the form of a proposition, whose proof relies on **Theorem 3.10**. After that, the "*concrete*" algorithm computing the RPA is given. Finally, we focus on some elements of the Coq formalization.

3.4.1 ▶ Division of rigorous polynomial approximations

For $f, g \in \mathcal{C}(I)$ with g nonvanishing over I, the quotient f/g is the unique root of $\mathbf{F} : h \mapsto gh - f$. Let h° be a candidate approximation given by the approximation step. Constructing the Newton-like operator $\mathbf{T}$ requires an approximation $\mathbf{A}$ of $(\mathcal{D}\mathbf{F}_{h^\circ})^{-1} : k \mapsto k/g$. For that purpose, suppose $w \approx 1/g \in \mathcal{C}(I)$ is also given by an oracle, and define:

$$\mathbf{T} \cdot h = h - w(gh - f). \tag{3.6}$$

The next proposition gives an upper bound for $\|h^\circ - f/g\|$.

Proposition 3.21 *Let $f, g, h^\circ, w \in \mathcal{C}(I)$, and $\mu, b \in \mathbb{R}_+$ such that:*

(3.21 i) $\|w(gh^\circ - f)\| \leqslant b$, **(3.21 ii)** $\|1 - wg\| \leqslant \mu$, **(3.21 iii)** $\mu < 1$.

Then g does not vanish over I and $\|h^\circ - f/g\| \leqslant \frac{b}{1-\mu}$.

Proof. Conditions **(3.21 ii)** and **(3.21 iii)** imply that $\mathbf{T}$ (Equation **(3.6)**) is contracting over $\mathcal{C}(I)$ with ratio μ. The radius $r := \frac{b}{1-\mu}$ makes the ball $\bar{B}(h^\circ, r)$ strongly stable with offset b **(3.21 i)**, since $b + \mu r = r$. Therefore, h^* is the (global) unique root of $\mathbf{F}$, and $\|h^\circ - h^*\| \leqslant r$. Finally, w and g do not vanish because $\|1 - wg\| \leqslant \mu < 1$. Hence, $h^* = f/g$ over I. □

Algorithm RpaDiv$(\mathbb{f}, \mathbb{g}, N_{\text{app}}, N_{\text{val}})$ computing a RPA for the division of $\mathbb{f}$ by $\mathbb{g}$ requires two extra parameters:

- a degree N_{app} for the degree of the approximation polynomial $h^\circ \approx f/g$;
- a degree N_{val} for the *witness* polynomial $w \approx 1/g$, need by **Proposition 3.21**.

Choosing the validation degree N_{val} is a trade-off between accuracy (w must be accurate enough to guarantee that μ is sufficiently small to yield a tight error bound) and efficiency (a large N_{val} increases the computation time of the validation step). It is totally independent of the approximation degree N_{app}.

Remark 3.22 *In the case of Chebyshev models, one can choose to impose $N_{\text{app}} = N_{\text{val}}$ and save time by evaluating f and g over the Chebyshev grid only once during the computation of h° and w. When using a global fixed degree N_{glob}, one can define a division operator on RPAs:*

$$\mathbb{f} \boxslash \mathbb{g} := \text{RpaDiv}(\mathbb{f}, \mathbb{g}, N_{\text{glob}}, N_{\text{glob}}),$$

which may return an exceptional value (or an RPA with infinite remainder) if the validation step fails.

Proposition 3.23 (Complexity of division of RPAs) *The overall complexity in interval operations of $\mathbb{f} \boxslash \mathbb{g}$ with fixed degree N_{glob} is determined by the 3 multiplications of RPAs involved in* RpaDiv. *In the common case of quadratic multiplication (e.g., in monomial or Chebyshev basis with naive multiplication), this requires $\mathcal{O}({N_{\text{glob}}}^2)$ interval arithmetic operations.*

In ChebValid, the division procedure `chebmodel_div` is implemented as the multiplication of $\mathbb{f}$ by the inverse of $\mathbb{g}$. The inverse (which is a particular case of division, with $f = 1$) is implemented by `chebmodel_inverse`. The validation step (corresponding to **Proposition 3.21**) is implemented by the auxiliary procedure `mpfr_chebpoly_inverse_validate`. In the future, the code must be updated to perform the division of RPAs in a single step (as presented above), to gain efficiency and accuracy.

Algorithm 3.5 RpaDiv$(\mathbb{f}, \mathbb{g}, N_{\text{app}}, N_{\text{val}})$ – Division of RPAs

Input: RPAs $\mathbb{f}, \mathbb{g}$, approximation degree N_{app} and validation degree N_{val}.
Output: RPA $\mathbb{h}$ of degree N_{app} for the division of $\mathbb{f}$ by $\mathbb{g}$.

1: $f \leftarrow \text{mid}(\mathbb{f})$ and $g \leftarrow \text{mid}(\mathbb{g})$
2: Compute a degree N_{app} approximation h° of f/g (e.g., using Chebyshev interpolation)
3: Compute a degree N_{val} approximation w of $1/g$
4: $\mathbb{h}^\circ \leftarrow (h^\circ, 0)$ and $\mathbb{w} \leftarrow (w, 0)$
5: $\mu \leftarrow \text{mag}\,([\![\mathbb{1} \boxminus \mathbb{w} \boxtimes \mathbb{g}]\!])$
6: **if** $\mu < 1$ **then**
7: $\quad b \leftarrow \text{mag}\,([\![\mathbb{w} \boxtimes (\mathbb{g} \boxtimes \mathbb{h}^\circ \boxminus \mathbb{f})]\!])$
8: $\quad \mathbb{\mu} \leftarrow [\mu]$ and $\mathbb{b} \leftarrow [b]$
9: $\quad \mathbb{r} \leftarrow \mathbb{b} \oslash (\mathbb{1} \ominus \mathbb{\mu})$
10: $\quad \mathbb{h} \leftarrow (h^\circ, \bar{r})$
11: $\quad$ **return** $\mathbb{h}$
12: **else**
13: $\quad$ **return** "Fail"
14: **end if**

Coq formalization of division

In our Coq development, the division of RPAs (a.k.a. `Model`) is implemented as follows:

```
Definition mdiv_aux (F G H W: Model): Model :=
 let K1 := 1-W*G in
 let K2 := W*(G*H - F) in
 match mag (mrange K1), mag (mrange K2) with
 | Some mu, Some b when is_lt mu 1 => {| pol := pol H; rem := rem H + sym (b/(1-mu)) |}
 | _ => mbot
 end.

Definition mdiv n (F G: Model): Model :=
  let p, q := mcf F, mcf G in
  mdiv_aux F G (mfc (interpolate n (fun x => beval p x / beval q x)))
               (mfc (interpolate n (fun x => 1 / beval q x))).
```

Since this Coq framework defines `Model` with error intervals rather than error bounds, `mdiv_aux` returns `(rem H + sym (b/(1-mu)))`. The interval `(sym (b/(1-mu)))` is equal to $[-r, r]$ with $r = \frac{b}{1-\mu}$,

whereas `(rem H)` is zero, since `H` is instantiated by `(mcf _)` in `mdiv`.

Note that we use the trivial model `mbot={|pol:=[];rem:=bot|}` as a default value, when the concrete computations fail to validate the guess of the oracle (either because this guess is just wrong, or because of over-approximations in the computations).

The correctness lemmas associated to these functions are:

```
Lemma rmdiv_aux F f G g H h W w:
  mcontains F f → mcontains G g → mcontains H h → mcontains W w →
  mcontains (mdiv_aux F G H W) (f/g).

Lemma rmdiv n F f G g: mcontains F f → mcontains G g → mcontains (mdiv' n F G) (f/g).
```

Lemma `rmdiv_aux` relies on the following formalization of **Proposition 3.21**. Since continuity was not used in that proposition, no such hypothesis appears in the Coq statement.

```
Lemma div.newton (f g h w : R → R) mu b :
  (∀ t, I t → Rabs (1 - w t * g t) ⩽ mu) →
  (∀ t, I t → Rabs (w t * (g t * h t - f t)) ⩽ b) →
  0 ⩽ mu < 1 → 0 ⩽ b →
  ∀ t, I t → Rabs (h t - f t / g t) ⩽ b / (1 - mu).
```

3.4.2 ▶ Square root of a rigorous polynomial approximation

Let $f \in \mathcal{C}(I)$ be strictly positive over I. The square root $\sqrt{f}$ is one of the two roots of the quadratic equation $\mathbf{F} \cdot h := h^2 - f = 0$ (the other being $-\sqrt{f}$). Let h° be a candidate approximation. Since $\mathcal{D}\mathbf{F}_h : k \mapsto 2hk$, one also needs an approximation $w \approx 1/(2h^\circ) \approx 1/(2\sqrt{f}) \in \mathcal{C}(I)$ in order to define $\mathbf{A} : k \mapsto wk$, approximating $(\mathcal{D}\mathbf{F}_{h^\circ})^{-1}$. Then:

$$\mathbf{T} : \quad h \quad \mapsto \quad h - w(h^2 - f).$$

The next proposition computes an upper bound for $\|h^\circ - \sqrt{f}\|$.

Proposition 3.24 *Let $f, h^\circ, w \in \mathcal{C}(I)$, $\mu_0, \mu_1, b \in \mathbb{R}_+$ and $t_0 \in I$ such that:*

(3.24 i) $\left\| w\left(h^{\circ 2} - f\right)\right\| \leqslant b$, **(3.24 ii)** $\|1 - 2wh^\circ\| \leqslant \mu_0$, **(3.24 iii)** $\|w\| \leqslant \mu_1$,

(3.24 iv) $\mu_0 < 1$, **(3.24 v)** $(1-\mu_0)^2 - 8b\mu_1 \geqslant 0$, **(3.24 vi)** $w(t_0) > 0$.

Then $f > 0$ over I and $\left\|h^\circ - \sqrt{f}\right\| \leqslant r^*$ where $r^* := \frac{1-\mu_0-\sqrt{(1-\mu_0)^2-8b\mu_1}}{4\mu_1}$.

Proof. First, since $\|1 - 2wh^\circ\| \leqslant \mu_0 < 1$ (by **(3.24 ii)** and **(3.24 iv)**) and $w(t_0) > 0$ **(3.24 vi)**, w and h° are strictly positive over I, by continuity. Using **(3.24 iii)**, $\mu_1 > 0$.

If $b = 0$, then $r^* = 0$ and $h^\circ = \sqrt{f}$ over I, because $w(h^{\circ 2} - f) = 0$ **(3.24 i)** and $w, h^\circ > 0$. Hence the conclusion holds.

From now on, we assume $b > 0$. $\mathbf{T}$ is Lipschitz of ratio $\mu(r) := \mu_0 + 2\mu_1 r$ over $\bar{B}(h^\circ, r)$ for any $r \in \mathbb{R}_+$, because:

$$\mathbf{T} \cdot h_1 - \mathbf{T} \cdot h_2 = (h_1 - h_2) - w(h_1^2 - h_2^2) = \left[(1 - 2wh^\circ) + w(h^\circ - h_1) + w(h^\circ - h_2)\right](h_1 - h_2).$$

Therefore, satisfying $b + \mu(r)r \leqslant r$ is equivalent to the quadratic inequality:

$$2\mu_1 r^2 + (\mu_0 - 1)r + b \leqslant 0. \tag{3.7}$$

Condition (3.24 v) implies that (3.7) admits solutions, and r^* is the smallest one. Moreover, since $b, \mu_1 > 0$, we get $r^* > 0$, so that $b + \mu(r^*)r^* = r^*$ also implies $\mu(r^*) < 1$.

Now, all the assumptions of **Theorem 3.10** are fulfilled. Hence, $\mathbf{T}$ has a unique fixed point h^* in $\bar{B}(h^\circ, r^*)$. To obtain $h^* = \sqrt{f}$ over I, it remains to show that $h^* > 0$. This follows from $w > 0$ and:

$$\|1 - 2wh^*\| \leqslant \|1 - 2wh^\circ\| + \|2w(h^* - h^\circ)\| \leqslant \mu_0 + 2\mu_1 r^* = \mu(r^*) < 1. \qquad \square$$

Similarly to **Algorithm** RpaDiv, RpaSqrt takes two degrees N_{app} and N_{val}, for the approximation and validation steps, respectively, and **Remark 3.22** still applies in the case of Chebyshev models. If the degree N_{glob} is globally fixed, one can define an operator for the square root:

$$\sqrt{\square}(\mathbb{f}) := \text{RpaSqrt}(\mathbb{f}, N_{\text{glob}}, N_{\text{glob}}).$$

Proposition 3.25 (Complexity of square root of RPA) *The overall complexity in interval operations of $\sqrt{\square}(\mathbb{f})$ with fixed degree N_{glob} is determined by the 3 multiplications of RPAs involved in* RpaSqrt. *In the common case of quadratic multiplication (e.g., in monomial or Chebyshev basis with naive multiplication), this requires $\mathcal{O}({N_{\text{glob}}}^2)$ interval arithmetic operations.*

In ChebValid, **Algorithm** RpaSqrt is implemented by `chebmodel_sqrt`, relying on the auxiliary procedure `mpfr_chebpoly_sqrt_validate`.

Algorithm 3.6 RpaSqrt$(\mathbb{f}, N_{\text{app}}, N_{\text{val}})$ – Square root of a RPA

Input: RPA $\mathbb{f}$, approximation degree N_{app} and validation degree N_{val}.
Output: RPA $\mathbb{h}$ of degree N_{app} for the square root of $\mathbb{f}$.

1: $f \leftarrow \text{mid}(\mathbb{f})$
2: Compute a degree N_{app} approximation h° of $\sqrt{f}$ (e.g., using Chebyshev interpolation)
3: Compute a degree N_{val} approximation w of $1/(2h^\circ)$
4: $\mathbb{h}^\circ \leftarrow (h^\circ, 0)$ and $\mathbb{w} \leftarrow (w, 0)$
5: $\mu_0 \leftarrow \text{mag}\,([\![\mathbb{1} \boxminus \mathbb{2} \boxdot \mathbb{w} \boxtimes \mathbb{h}^\circ]\!])$ and $\mathbb{\mu}_0 \leftarrow [\mu_0]$
6: $\mu_1 \leftarrow \text{mag}\,([\![\mathbb{w}]\!])$ and $\mathbb{\mu}_1 \leftarrow [\mu_1]$
7: $b \leftarrow \text{mag}\,([\![\mathbb{w} \boxtimes (\mathbb{h}^\circ \boxtimes \mathbb{h}^\circ \boxminus \mathbb{f})]\!])$ and $\mathbb{b} \leftarrow [b]$
8: $\mathbb{\delta} \leftarrow (\mathbb{1} \ominus \mathbb{\mu}_0)^{②} - [8] \otimes \mathbb{b} \otimes \mathbb{\mu}_1$
9: Choose some $t_0 \in I$ and evaluate $\mathbb{y} \leftarrow \mathbb{w}([t_0])$
10: **if** $\overline{\mu}_0 < 1$ **and** $\underline{\delta} \geqslant 0$ **and** $\underline{y} > 0$ **then**
11: $\quad \mathbb{r} \leftarrow (\mathbb{1} \ominus \mathbb{\mu}_0 \ominus \sqrt{\mathbb{\delta}}) \oslash ([4] \otimes \mathbb{\mu}_1)$
12: $\quad \mathbb{h} \leftarrow (h^\circ, \bar{r})$
13: $\quad$ **return** $\mathbb{h}$
14: **else**
15: $\quad$ **return** "Fail"
16: **end if**

Coq formalization of square root

The Coq functions computing the square root of a RPA, together with the corresponding correctness lemmas, are listed below.

```
Let msqrt_aux (F H W: Model) (x: II): Model :=
  let Wx := meval W x in
  if ~~ (is_lt lo x && is_lt x hi && is_lt 0 Wx) then mbot else
  let K1 := 1 - 2*W*H in
  let K2 := W*(H*H-F) in
  match mag (mrange K1), mag (mrange W), mag (mrange K2) with
  | Some mu0, Some mu1, Some b =>
    let delta := (1 - mu0)^2 - 8*b*mu1 in
    let rmin := (1 - mu0 - sqrt delta)/(4*mu1) in
    let mu := mu0 + 2*mu1*rmin in
    if is_lt mu0 1 && is_lt 0 delta && is_lt mu' 1 then
    {| pol := pol H; rem := rem H + sym rmin' |}
    else mbot
  | _ => mbot
  end.

Let msqrt n (F: Model): Model :=
  let p: seq FF := mcf F in
  let h: seq FF := interpolate n (fun x => sqrt (beval p x)) in
  msqrt_aux M (mfc h) (mfc (interpolate n (fun x => 1/(2*beval h x)))) ((lo+hi)/2).

Lemma rmsqrt_aux (F H W: Model) (X: II) (f h w : R → R) (x: R):
  mcontains F f → mcontains H h → mcontains W w → contains X x → lo⩽x⩽hi →
  (∀ x, lo⩽x⩽hi → continuity_pt w x) →
  mcontains (msqrt_aux F H W X) (sqrt f).

Lemma rmsqrt n F f: mcontains F f → mcontains (msqrt' n F) (sqrt f).
```

Lemma `rmsqrt_aux` makes use of the following formalization of **Proposition 3.24**:

```
Lemma sqrt.newton (f h w : R → R) mu0 mu1 b :
  (∀ t, I t → Rabs (1 - 2 * w t * h t) ⩽ mu0) →
  (∀ t, I t → Rabs (w t) ⩽ mu1) →
  (∀ t, I t → Rabs (w t * ((h t)^2 - f t)) ⩽ b) →
  0 ⩽ mu0 < 1 → 0 < mu1 → 0 ⩽ b → 0 ⩽ delta b mu0 mu1 → mu0 + 2 * mu1 * rmin b mu0 mu1 <
    1 →
  (∀ t1 t2 t3, t1 ⩽ t2 ⩽ t3 → I t1 → I t3 → I t2) →
  (∀ t, I t → continuity_pt w t) → (∃ t0, I t0 /\ w t0 > 0) →
  ∀ t, I t → Rabs (h t - sqrt (f t)) ⩽ rmin b mu0 mu1.
```

Remark 3.26 *Contrary to the case of division where continuity was not needed at all, it is here used for w. Therefore,* `sqrt.newton` *requires w to be continuous over I.*

3.5 Examples using the arithmetic on rigorous polynomial approximations

Checking positivity of functions using elementary RPAs operations

Rigorously checking positivity of univariate functions is a well-studied problem, with many applications (e.g., close to our interest, bounding the approximation error for polynomial approximations [54]). Here, we present an elementary technique using arithmetic operations on RPAs.

Consider $f \in \mathrm{Y}^1$. Recall from Section 2.2.4 that if f does not vanish over $[-1,1]$, then $g := 1/f \in \mathrm{Y}^1$. Hence, one can compute a polynomial approximation g° of g, together with a rigorous bound $\varepsilon \geqslant \|1 - g^\circ f\|_{\mathrm{Y}^1}$. For any $x \in [-1,1]$, $f(x) = 0$ implies:

$$1 = |1 - g^\circ(x)f(x)| \leqslant \|1 - g^\circ f\|_\infty \leqslant \|1 - g^\circ f\|_{\mathrm{Y}^1} \leqslant \varepsilon.$$

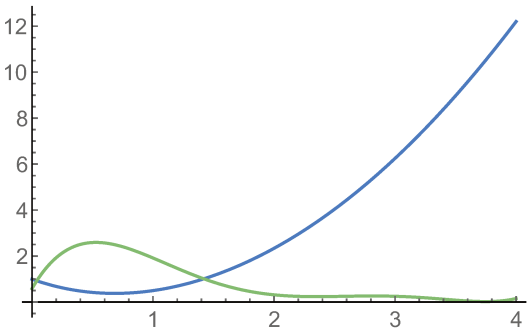
Hence, if $\varepsilon < 1$, then f cannot vanish over $[-1,1]$. By continuity of f, it is sufficient to check in addition that $f(x) \geqslant 0$ for some $x \in [-1,1]$. This sketch of proof is implemented with RPAs by Algorithm RpaPositivityCheck, for which two examples are given below.

Algorithm 3.7 RpaPositivityCheck$(\mathbb{f}, N_{\mathrm{app}})$ – Rigorous positivity check

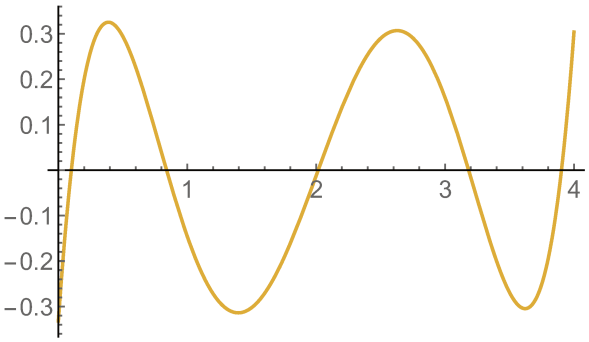
Input: an RPA $\mathbb{f}$ and an approximation degree N_{app}.
Output: **true** or **false**, with **true** asserting that $f > 0$ over $[-1,1]$ for any $f \in \mathbb{f}$.

1: Choose an $x \in [-1,1]$ and compute $\mathbb{y} \leftarrow \mathbb{f}([x])$
2: **if** $\mathbb{y} \leqslant 0$ **then**
3: **return false**
4: **end if**
5: Compute a degree N_{app} approximation g° of $1/f$ using Chebyshev interpolation
6: $\mathbb{g} \leftarrow (g^\circ, 0)$
7: $\varepsilon \leftarrow \mathrm{mag}(\llbracket \mathbb{1} \boxminus \mathbb{g} \otimes \mathbb{f} \rrbracket_{\mathrm{Y}^1})$
8: **if** $\varepsilon < 1$ **then**
9: **return true**
10: **else**
11: **return false**
12: **end if**

Example 3.27 *Consider again the function* $f : x \mapsto \frac{1}{1+x} - x + x^2$ *from* ***Example 1.23*** *in* ***Chapter 1****. An RPA* $\mathbb{f}$ *for* f *is easily computed using division defined in* ***Section 3.4.1****. Now,* ***Algorithm*** RpaPositivityCheck$(\mathbb{f}, 5)$ *returns* ***true****, which means that a degree* 5 *approximation is sufficient to rigorously assert the positivity of* f *over* $[0,4]$ *(which was rescaled to* $[-1,1]$ *by an affine change of variable).* ***Figure 3.3a*** *plots* f *and the computed* $g^\circ \approx 1/f$*. The "error function"* $1 - g^\circ f$ *has a* Y^1 *norm bounded by* 0.9261*, and is depicted in* ***Figure 3.3b****.*



(a) Function f (in blue) and computed degree 5 approximation $g^\circ \approx 1/f$ (in green).



(b) Function $x \mapsto 1 - g^\circ(x)f(x)$.

■ **Figure 3.3**: Positivity of $f : x \mapsto \frac{1}{1+x} - x + x^2$ rigorously checked by RpaPositivityCheck$(\mathbb{f}, 5)$.

■ **Example 3.28** *In Section 1.3, the Taylor remainder of the* cos *function over* $[-1, 1]$ *(Equation (1.3)) was used to illustrate the limits of interval subdivision techniques:*

$$f_n(x) = \cos(x) - \sum_{i=0}^{2n} (-1)^i \frac{x^{2i}}{(2i)!}, \qquad n \geqslant 0,$$

Now, using RPAs, we are able to give efficient bounds for $|f_n|$*. First, we construct a Chebyshev model* $\mathbb{f}_n$ *for* f_n*. This requires to construct one for the* cos *function, which is possible by anticipating the framework given in the following chapter. Then, take a candidate upper bound* η *for* $|f_n|$*, for example by upward rounding a numerical evaluation of* f_n *at* 1*. Finally, it suffices to check the positivity of* $\varepsilon - f$ *and* $\varepsilon + f$ *over* $[-1, 1]$*, using* RpaPositivityCheck*.*

Using the C implementation, we manage to prove, for instance, that $\|f_{10}\|_\infty \leqslant 8.8807 \cdot 10^{-22}$*, using degree 40 approximations, which is to be compared with* $\mathbb{f}_{10}([1]) = [-8.880698780 \cdot 10^{-22}, -8.880698779 \cdot 10^{-22}]$*. Clearly, RPA methods outperform naive interval branch and bound techniques on that example.*

■ Playing with approximations of the absolute value function

Consider the function $f_\varepsilon : x \mapsto \sqrt{\varepsilon + x^2}$ over $[-1, 1]$, with $\varepsilon > 0$. When $\varepsilon \to 0$, f_ε converges uniformly to the absolute value function $x \mapsto |x|$ (which is not analytic at 0), with:

$$|f(x) - |x|| = \left|\sqrt{\varepsilon + x^2} - \sqrt{x^2}\right| = \left|\frac{\varepsilon}{\sqrt{\varepsilon + x^2} + \sqrt{x^2}}\right| \leqslant \sqrt{\varepsilon}, \qquad \text{for all } x \in [-1, 1]. \qquad (3.8)$$

Rigorous uniform approximations. Approximating f_ε with polynomials becomes harder for small ε, due to the complex singularities $\pm i\sqrt{\varepsilon}$ getting closer to the interval $[-1, 1]$. Nevertheless, Chebyshev interpolation still works and our implementation computes rigorous approximations as accurate as desired (see **Figure 3.4b**), of exponential convergence with ratio determined by ε. Note that for too small a degree, the computed approximation of the square root is too far from the solution, and the a posteriori validation returns an infinite remainder.

In order to provide a comparison with CoqApprox's Taylor models, we used the tactic `interval with (i_depth 1, i_bisect_taylor x N, i_prec p)` to build a Taylor model of degree N

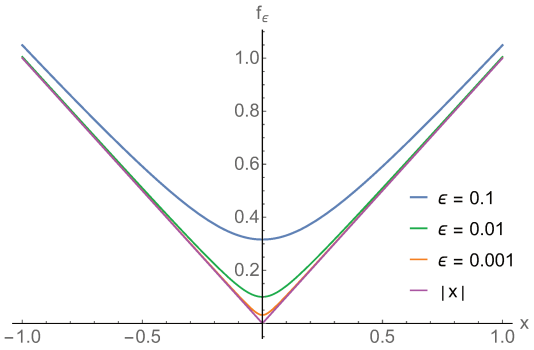
with precision p. Timings given in **Table 3.4c** reveal a significant advantage of our implementation (there we use $\varepsilon = 2$ to avoid convergence issues of Taylor models). Concerning accuracy, our experiments tend to show that when $\varepsilon \leqslant 1$, CoqApprox fails to compute *converging* Taylor models. Indeed, even with large L, a goal like:

```
Goal Fail : ∀ x : R, -1 ⩽ x ⩽ 1 → sqrt (1/100+x*x) ⩽ L
```

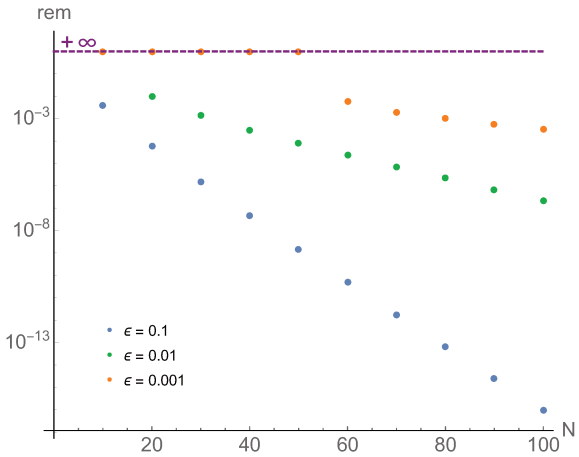
is not solved when the degree N becomes too large, probably indicating that the Taylor models *diverge* due to complex singularities inside the unit disk. Note that the `interval` tactic can solve this goal, but only by resorting to subdivision techniques.

Error bounding. We want to bound $|f_\varepsilon(x) - |x||$ for $x \in [-1, 1]$ without making use of any symbolic manipulation like **(3.8)**. At first glance, one can choose to use the rigorous approximations over $[-1, 1]$ obtained previously, and evaluate $f_\varepsilon(x) - x$ (resp. $f_\varepsilon(x) + x$) over $[0, 1]$ (resp. $[-1, 0]$) using Clenshaw algorithm. However, even if the approximations are quite good, this evaluation strategy gives huge overestimations because $[0, 1]$ and $[-1, 0]$ are not small intervals.

Instead, we compute separately two approximations for f_ε: one over $[0, 1]$ and one over $[-1, 0]$, and we evaluate $f_\varepsilon(x) - x$ (resp. $f_\varepsilon(x) + x$) over $[1, 0]$ (resp. $[-1, 0]$) using the Chebyshev `range` function. This approach yields bounds that are rather close to the optimal $\sqrt{\varepsilon}$ (see **Figure 3.4d**). However, this does not allow for arbitrary accuracy: a subdivision procedure would be necessary here.



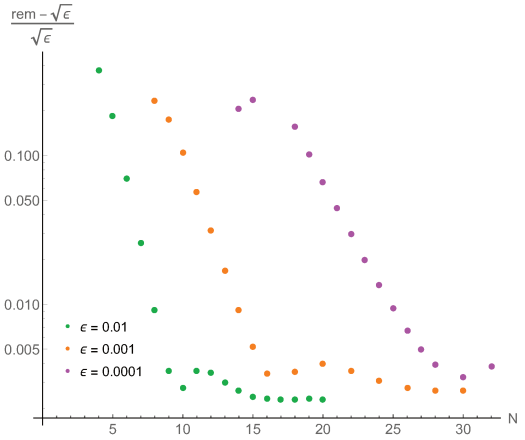
(a) Functions f_ε and $x \mapsto |x|$ over $[-1, 1]$.



(b) Magnitude of the remainders of degree-N Chebyshev models approximating f_ε over $[-1, 1]$.

N	time (in seconds)	
	Chebyshev	CoqApprox
10	0.11	0.10
20	0.16	0.12
30	0.22	0.14
40	0.31	0.20
50	0.42	0.29
60	0.56	0.44
70	0.71	0.64
80	0.89	0.93
90	1.08	1.33
100	1.31	1.84
120	1.80	3.34
140	2.43	5.60
160	2.98	8.89
180	3.86	13.47
200	4.75	19.71

(c) Timings of degree-N models for f_2.



(d) Overapproximation ratio of the remainder of the degree-N Chebyshev model for $f_\varepsilon - 1$ over $[0, 1]$, compared to the optimal bound $\sqrt{\varepsilon}$.

■ **Figure 3.4**: Approximating functions f_ε and $x \mapsto |x|$ with Chebyshev models.

Part II

Chebyshev Models for D-finite functions (and more)

Chebyshev Models for Solutions of Linear Ordinary Differential Equations

4

Das Wesen der Mathematik liegt gerade in ihrer Freiheit.
— Georg Cantor
Aus dem Paradies, das Cantor uns geschaffen, soll uns niemand vertreiben können.
— David Hilbert, Über das Unendliche

Solutions of Linear Ordinary Differential Equations (LODEs) are ubiquitous in modeling and solving common problems. Examples include elementary and special functions evaluation, manipulation or plotting, numerical integration, or locally solving nonlinear problems using linearizations.

This chapter takes over the journal article "Validated and numerically efficient Chebyshev spectral methods for linear ordinary differential equations" [38] published in 2018 in *ACM Transactions on Mathematical Software*. Therefore, the pronoun *we* stands for Nicolas Brisebarre, Mioara Joldes and myself. Our contribution is an efficient algorithm for computing rigorous polynomial approximations for solutions of LODEs. More specifically, we deal with the following problem:

Problem 4.1 *Let r be a positive integer, $\alpha_0, \alpha_1, \ldots, \alpha_{r-1}$ and γ continuous functions over $[-1,1]$. Consider the LODE*

$$f^{(r)}(t) + \alpha_{r-1}(t) f^{(r-1)}(t) + \cdots + \alpha_1(t) f'(t) + \alpha_0(t) f(t) = \gamma(t),\ t \in [-1,1], \tag{4.1}$$

together with conditions uniquely characterizing the solution:

a) For an initial value problem *(IVP), consider:*

$$\mathbf{\Lambda} \cdot f := (f(t_0), f'(t_0), \ldots, f^{(r-1)}(t_0)) = (v_0, v_1, \ldots, v_{r-1}) \tag{1a}$$

for given $t_0 \in [-1,1]$ and $(v_0, v_1, \ldots, v_{r-1}) \in \mathbb{R}^r$.

b) For a generalized boundary value problem *(BVP), conditions are given by r linearly independent linear functionals $\lambda_i : \mathcal{C}^{r-1} \to \mathbb{R}$:*

$$\mathbf{\Lambda} \cdot f := (\lambda_0(f), \dots, \lambda_{r-1}(f)) = (\ell_0, \dots, \ell_{r-1}) \tag{1b}$$

for given $(\ell_0, \dots, \ell_{r-1}) \in \mathbb{R}^r$.

Given an approximation degree $N_{\text{app}} \in \mathbb{N}$, find the coefficients of a polynomial $f^\circ(t) = \sum_{n=0}^{N_{\text{app}}} c_n T_n(t)$ written in Chebyshev basis (T_n), together with a tight and rigorous error bound η such that $\|f^\circ - f^*\|_\infty := \sup_{t\in[-1,1]} |f^\circ(t) - f^*(t)| \leqslant \eta$, where f^* is the exact solution of the IVP/BVP.

Remark 4.2 *On the practical side, the coefficients α_i and the initial conditions v_i appearing in **Problem 4.1** are represented by RPAs $\mathfrak{a}_i$ and intervals $\mathbb{v}_i$, supposed to be rather thin. Since we only deal with the linear case here, working with large intervals $\mathbb{v}_i$ for the initial conditions does not make the problem much harder: one just needs to compute a basis of solutions $\mathbb{f}_i$ for canonical initial conditions, and return $\mathbb{v}_0 \boxdot \mathbb{f}_0 \boxplus \cdots \boxplus \mathbb{v}_{r-1} \boxdot \mathbb{f}_{r-1}$.*

Related works. Integrating ODEs is a central topic for rigorous numerics, and many tools were developed toward this end. Interval arithmetic [174] gave rise to numerous interval methods for linear or nonlinear IVPs. Some formal proof implementations should also be acknowledged, such as certified ODE solving procedures [120], leading to a formal proof [119] of the already rigorously proved Lorenz attractor [246], or the work [28] providing certified bounds for the rounding errors occurring during reference implementations of Runge-Kutta schemes.

However, the so-called *wrapping effect* discussed in **Section 1.3** (see [185]) led to the development of *higher-order methods*, including the rigorous polynomial approximations (RPAs), presented in the previous chapter, and already advocated in the 1980s for such problems [136, 199]. *Taylor models*, introduced some years later in [164], were used in several differential problems [23, 163, 167, 181]. Due to limited convergence properties of Taylor expansions, focus was also put on alternative approximation tools, such as Chebyshev expansions [135, 19, 153, 75].

More specifically, the framework developed in this chapter follows the spirit of [19], where the authors proposed a fully automated algorithm using *a posteriori* validation method, based on convergent Neumann series of linear operators in the Banach space of continuous functions $(\mathcal{C}^0, \|\cdot\|_\infty)$. This allows for efficient RPA solutions of LODEs with polynomial coefficients, also called D-finite functions, presented in **Section 2.1**. While in the rigorous numerics field only ad hoc methods are usually described, the authors of [19] initiated a computer algebra-like approach by providing a fully documented algorithm with detailed complexity estimates. This chapter extends this complexity study to the framework of Newton-like validation methods (**Section 3.3** for **Problem 4.1**.

Overview of our approach and main results. We develop an efficient algorithm for solving **Problem 4.1** when the coefficients α_j and the right hand side γ are represented by Chebyshev models, which can be done to an arbitrary accuracy under mild regularity assumptions, as detailed in **Chapters 2** and **3**.

We first give in **Section 4.1** a classical integral reformulation of (4.1). This has the advantage of directly producing a compact operator, yielding appropriate fast convergence results of the solution of truncated linear systems to the exact one (see **Theorem 4.11**). Moreover, we prove an important property from an algorithmic point of view: this compact operator has an *almost-banded matrix representation* when Equation (4.1) has polynomial coefficients. This leads to the formulation of the following subproblem, where for the sake of simplicity, we focus on the case of an IVP. Note also that approximations over other real or complex segments (in Chebyshev basis adapted to the segment) are reduced to approximations on $[-1,1]$ by means of the affine change of variable $t \mapsto (1-t)a/2 + (1+t)b/2$, mapping $[-1,1]$ to the segment $[a,b]$.

Problem 4.3 *Let $a_0, a_1, \ldots, a_{r-1}, g \in \mathbb{R}[t]$. Consider the LODE*

$$f^{(r)}(t) + a_{r-1}(t)f^{(r-1)}(t) + \cdots + a_1(t)f'(t) + a_0(t)f(t) = g(t), \quad t \in [-1,1], \tag{4.2}$$

over $[-1,1]$ together with initial conditions at $t_0 = -1$:

$$f(t_0) = v_0, \quad f'(t_0) = v_1, \quad \ldots, \quad f^{(r-1)}(t_0) = v_{r-1}.$$

Given $N_{\text{app}} \in \mathbb{N}$, find the coefficients of $f^\circ(t) = \sum_{n=0}^{N_{\text{app}}} c_n T_n(t)$ and a tight and rigorous error bound η such that $\|f^\circ - f^*\|_\infty \leqslant \eta$.

Remark 4.4 *Note that in this problem we focus on the case $t_0 = -1$ for technical reasons explained in Section 4.3.1, but our results remain valid for any $t_0 \in [-1,1]$.*

According to the general framework of a posteriori validation methods in **Section 3.3**, this problem is solved with the following steps:

Step 1. An approximate solution is necessary. This can be provided by the user, that is, computed by some numerical algorithm of choice (such as that of **[191]** or **[19]**). For completeness of our implementation, we propose a linear (with respect to the approximation degree) time approximation algorithm, which combines the classical integral reformulation mentioned above and the algorithm for almost-banded linear systems from **[191]**, recalled in **Section 4.2**.

Then, we develop a new variant of this algorithm, which is efficient (in many practical cases) for obtaining the approximate inverse operator $\mathbf{A}$ involved in the definition of the Newton-like operator, used in the next step.

Step 2. A new algorithm based on the Banach fixed-point theorem is proposed in **Section 4.3**. It provides the rigorous approximation error bound required by **Problem 4.3**.

In particular, for a *fixed* given LODE, our validation algorithm runs in linear time, in terms of basic arithmetic operations, with respect to the degree N_{app} of the approximation to be validated.

Then, we generalize this method in **Section 4.4** in two directions:

- when the coefficients α_j are not polynomials anymore, but functions in $\mathcal{U}^1$ represented by Chebyshev models;
- and when the conditions are generalized boundary conditions (1b).

This allows us to construct Chebyshev models for a quite large class of functions, starting from $\mathcal{H}_0 = \mathbb{R}[t]$ and defining $\mathcal{H}_{i+1}$ as the solutions of **Problem 4.1** where all the $\alpha_j(t)$ and $\gamma(t)$ are in $\mathcal{H}_i$, or some closure of it under other elementary operations like inversion, square root, etc.

(see Section 3.4). In fact, if the $\alpha_j(t)$ and $\gamma(t)$ are rigorously approximated by Chebyshev models, then the generalized method gives us a Chebyshev model for the solution. Thus, a chain of recursive calls to the method can be used to approximate any function of $\mathcal{H} := \bigcup_i \mathcal{H}_i$.

Finally, we illustrate our approach with four different examples in Section 4.5. Concluding remarks and future directions are postponed at the end of Chapter 5. Note that all the algorithms introduced in this chapter are implemented in the ChebValid C library[1] presented in the previous chapter, and the code of the examples in Section 4.5 is also available in this repository.

4.1 Integral operator and its truncations

Due to the differentiation formula for Chebyshev polynomials (2.8), the differentiation operator on the Chebyshev coefficients is represented by a dense upper triangular matrix. It implies that a direct translation of the differential equation (4.1) into a linear problem produces a dense infinite-dimensional system of linear equations. Moreover it is ill-conditioned in the general case [99]. Hence, numerical algorithms to solve (4.1) using this method are neither efficient nor accurate. From the validation point of view, since the differentiation is not an endomorphism of Y^1 (some functions in Y^1 are *not* even differentiable), designing a topological fixed-point method directly from Equation (4.1) seems rather tedious.

One way to circumvent these limitations consists in transforming the differential equation (4.1) into an integral one. The indefinite integration operator has far better properties: first, it is an endomorphism of Y^1. Second, it has a sparse matrix representation in Y^1, (due to the integration formula (2.7)), and its conditioning is significantly better than that of the differential one [99]. Thus, one can expect more efficient and accurate numerical algorithms in this case. The following standard, but crucial proposition (see [145] or [260, Chap. 2] for a proof) establishes this transformation, which was already used in purely numerical works for LODEs (for example in [61]) as well as for validation purposes [19].

Proposition 4.5 *Let f be a function of class $\mathcal{C}^r$ over $[-1,1]$. Then f is a solution of the linear IVP problem (1a) if and only if $\varphi = f^{(r)} \in \mathcal{C}^0$ is solution of the Volterra integral equation:*

$$\varphi + \mathbf{K}\cdot\varphi = \psi \quad \textit{with} \quad (\mathbf{K}\cdot\varphi)(t) = \int_{t_0}^{t} k(t,s)\varphi(s)\mathrm{d}s, \quad t \in [-1,1], \tag{4.3}$$

where:

- *the kernel $k(t,s)$ is a bivariate continuous function given by:*

$$k(t,s) = \sum_{j=0}^{r-1} \alpha_j(t)\frac{(t-s)^{r-1-j}}{(r-1-j)!}, \quad (t,s) \in [-1,1]^2,$$

[1] https://gforge.inria.fr/projects/tchebyapprox/

- *the right hand side ψ is given by:*

$$\psi(t) = \gamma(t) - \sum_{j=0}^{r-1} \alpha_j(t) \sum_{k=0}^{r-1-j} v_{j+k} \frac{(t-t_0)^k}{k!}, \quad t \in [-1,1].$$

By a slight abuse of terminology, we shall call r the order of the integral operator $\mathbf{K}$.

Remark 4.6 *Proposition 4.5 can be applied to the polynomial case of Problem 4.3 by replacing α_j and γ with polynomials a_j and g. It produces an equivalent integral equation with a bivariate* polynomial *kernel $k(t,s)$ and a* polynomial *right hand side $\psi(t)$. This will be of first importance in Section 4.1.2 where we deal with the polynomial case.*

Remark 4.7 *Henceforth, as noted in Equation (4.3), the new unknown function is $\varphi = f^{(r)}$. Although a similar integral formulation for the unknown f is possible, this choice allows for the validation in Section 4.3 of numerical solutions for both f and its derivatives $f^{(i)}$, $i = 1, \ldots, r$, which is often required in validated dynamics cf. Example 4.5.4.*

Solving Equation (4.3) with numerical algorithms on computers relies most of the time [93, 96] on a reduction of this infinite-dimensional problem to a finite-dimensional one. In fact, usually, one approximately computes several coefficients of the Chebyshev expansion of the exact solution. This is often done based on approximations of the inverse operator. The question of which functional space the solution φ belongs to is of major importance both for the numerical approximation and the computation of the validated uniform error bound. In what follows we first recall the classical action of $\mathbf{K}$ on $(\mathcal{C}^0([-1,1]), \|\cdot\|_\infty)$, with a focus on Picard iteration. Then, in Section 4.1.2, we prove analogous properties in the $(\text{Ч}^1, \|\cdot\|_{\text{Ч}^1})$ space, based on operator iterations and truncations. This Banach space proves to be the natural framework to deal with Chebyshev coefficients without losing the link with the norm $\|\cdot\|_\infty$ (since $\|\cdot\|_\infty \leqslant \|\cdot\|_{\text{Ч}^1}$).

4.1.1 ▶ Inverse of $\mathbf{1}+\mathbf{K}$ in $(\mathcal{C}^0([-1,1]), \|\cdot\|_\infty)$

It is classical that in this Banach space the operators $\mathbf{K}$ and $\mathbf{1}+\mathbf{K}$ are bounded linear endomorphisms. For $n \in \mathbb{N}$, the operator $\mathbf{K}^n$ is a bounded linear operator with operator norm

$$\|\mathbf{K}^n\|_\infty \leqslant \frac{(2C)^n}{n!}, \text{ where } C := \sup_{-1\leqslant s,t\leqslant 1} |k(t,s)| < +\infty. \tag{4.4}$$

Picard iteration [145, 205] is a standard way to prove the invertibility of $\mathbf{1}+\mathbf{K}$ in $(\mathcal{C}^0([-1,1]), \|\cdot\|_\infty)$ and give an explicit form for $(\mathbf{1}+\mathbf{K})^{-1}$, by its Neumann series:

$$(\mathbf{1}+\mathbf{K})^{-1} = \sum_{n=0}^{+\infty} (-1)^n \mathbf{K}^n = \mathbf{1} - \mathbf{K} + \mathbf{K}^2 + \cdots + (-1)^n \mathbf{K}^n + \cdots.$$

This yields an explicit approximation process for the solution of (4.3):

$$\varphi_0 = \psi, \qquad \varphi_{n+1} = \psi - \mathbf{K}\cdot\varphi_n = \left(\sum_{k=0}^{n} (-1)^k \mathbf{K}^k\right)\cdot\psi, \; n \in \mathbb{N}.$$

Iterating the integral operator $\mathbf{K}$ can also be used for validation purposes, as presented for example in [19]. However, in our quasi-Newton validation context, the Banach space $(\mathcal{C}^0, \|\cdot\|_\infty)$ seems difficult to work with when considering multiplication, integration and truncation of Chebyshev series as operations on the coefficients.

4.1.2 ▶ Inverse of $\mathbf{1}+\mathbf{K}$ in $(\mathrm{Ч}^1, \|\cdot\|_{\mathrm{Ч}^1})$

In this section, we provide a concrete description of the action of the integral operator $\mathbf{K}$ on the Chebyshev coefficients of a function.

Remark 4.8 *Henceforth and until the end of Section 4.1, we exclusively consider the polynomial case given by (4.2). The results presented below could to some extent be generalized to the non-polynomial case (where all functions belong to $\mathrm{Ч}^1$), but this would require a more complicated two-variable approximation theory without being essential to the validation procedure of the general Problem 1a, presented in Section 4.4.*

Under this assumption, the kernel $k(t,s)$ is polynomial and hence we can decompose it in the Chebyshev basis according to the variable s:

$$k(t,s) = \sum_{j=0}^{r-1} b_j(t) T_j(s), \tag{4.5}$$

with $b_0, \dots, b_{r-1}$ polynomials written in the Chebyshev basis. Such an elementary procedure is described in **Algorithm** IntegralTransform. To implement it in a rigorous framework, one can use interval arithmetics or even rational arithmetics when the coefficients of the $a_j(t)$ are rationals.

If $\varphi \in \mathrm{Ч}^1$, then $\mathbf{K}\cdot\varphi$ is in $\mathrm{Ч}^1$ since

$$\|\mathbf{K}\cdot\varphi\|_{\mathrm{Ч}^1} = \left\| \sum_{j=0}^{r-1} b_j(t) \int_{t_0}^{t} T_j \varphi \mathrm{d}s \right\|_{\mathrm{Ч}^1} \leqslant 2B\|\varphi\|_{\mathrm{Ч}^1},$$
$$\text{with } B = \sum_{j=0}^{r-1} \|b_j\|_{\mathrm{Ч}^1} \geqslant C,$$

where we used (2.10), (2.12) and C was defined in Equation (4.4). This shows that $\mathbf{K}$, and hence $\mathbf{1}+\mathbf{K}$, are bounded linear endomorphisms of $\mathrm{Ч}^1$. However, we do not have for the moment any information about the invertibility of $\mathbf{1}+\mathbf{K}$ in $\mathrm{Ч}^1$. So far, its injectivity in $\mathrm{Ч}^1$ is established, because this operator was an isomorphism (hence injective) over the superspace $\mathcal{C}^0([-1,1]))$.

Matrix representation of $\mathbf{1}+\mathbf{K}$ in $\mathrm{Ч}^1$

The canonical matrix representation of a linear operator $\mathbf{M} : \mathrm{Ч}^1 \to \mathrm{Ч}^1$ is the infinite-dimensional matrix $M = (M_{ij})_{i,j\in\mathbb{N}}$, where $M_{ij} = c_i(\mathbf{M}\cdot T_j)$ is the ith Chebyshev coefficient of the image

Algorithm 4.1 IntegralTransform($\{a_j\}_{j=0}^{r-1}$) – Computation of the kernel $k(t,s)$

Input: order r and polynomials $a_j(t)$ ($j \in [\![0, r-1]\!]$) written in the Chebyshev basis.
Output: polynomials $b_j(t)$ ($j \in [\![0, r-1]\!]$) defining the kernel $k(t,s)$ as in (4.5).

▷ *Expand* $(t-s)^k = \sum_{\ell=0}^{k} \xi_{k\ell}(t) T_\ell(s)$ *for* $k \in [\![0, r-1]\!]$.
$\xi_{00}(t) \leftarrow 1$
for $k = 1$ **to** $r-1$ **do**
  for $\ell = 0$ **to** k **do** $\xi_{k\ell}(t) = 0$ **end for**
  for $\ell = 0$ **to** $k-1$ **do**
    $\xi_{k\ell}(t) \leftarrow \xi_{k\ell}(t) + t\xi_{k-1,\ell}(t)$
    $\xi_{k,\ell+1}(t) \leftarrow \xi_{k,\ell+1}(t) - \xi_{k-1,\ell}(t)/2$
    $\xi_{k,|\ell-1|}(t) \leftarrow \xi_{k,|\ell-1|}(t) - \xi_{k-1,\ell}(t)/2$
  end for
end for

▷ *Compute the* $b_j(t)$.
for $j = 0$ **to** $r-1$ **do**
  $b_j(t) \leftarrow 0$
  for $k = 0$ **to** $r-1$ **do**
    $b_j(t) \leftarrow b_j(t) + a_k(t)\xi_{r-1-k,j}(t)/(r-1-k)!$
  end for
end for

of T_j under $\mathbf{M}$. **Lemma 2.41** implies that the $\mathrm{Ч}^1$ operator norm of $\mathbf{M}$ is equal to the standard matrix 1-norm of M:

$$\|\mathbf{M}\|_{\mathrm{Ч}^1} = \sup_{j\in\mathbb{N}} \|\mathbf{M}\cdot T_j\|_{\mathrm{Ч}^1} = \sup_{j\in\mathbb{N}} \sum_{i\in\mathbb{N}} |c_i(\mathbf{M}\cdot T_j)| = \sup_{j\in\mathbb{N}} \sum_{i\in\mathbb{N}} |M_{ij}| =: \|M\|_1.$$

For ease of calculation, we use *two-sided* Chebyshev expansions in this section, by defining $T_{-n} = T_n$ for $n \geqslant 0$. This makes it possible to remove the absolute values in the indices appearing in the formulas for multiplication (2.6) and integration (2.7).

First, we express the polynomials b_j of degree d_j in the symmetric two-sided Chebyshev basis:

$$b_j = \sum_{-d_j \leqslant k \leqslant d_j} b_{j,k} T_k, \quad \text{with} \quad b_{j,k} = b_{j,-k}, \quad 0 \leqslant j < r.$$

For $i, j \in \mathbb{Z}$, we have $T_j T_i = (T_{i+j} + T_{i-j})/2$. Now, for $t \in [-1, 1]$, we have

$$\int_{t_0}^{t} T_j(s) T_i(s) \mathrm{d}s = \gamma_{iji}(t) - \gamma_{iji}(t_0),$$

with

$$\gamma_{ijk}(t) = -\frac{1}{4(i-j-1)} T_{k-j-1}(t) + \frac{1}{4(i-j+1)} T_{k-j+1}(t) - \frac{1}{4(i+j-1)} T_{k+j-1}(t) + \frac{1}{4(i+j+1)} T_{k+j+1}(t), \quad (4.6)$$

where by convention the terms for which the denominator vanishes are 0.

In particular, for $t_0 = -1$ and using $T_k(-1) = (-1)^k$ one obtains:

$$\gamma_{ijk}(-1) = -(-1)^{k+j}\left(\frac{j+1}{2(i^2-(j+1)^2)} + \frac{j-1}{2(i^2-(j-1)^2)}\right). \tag{4.7}$$

Let $j \in [\![0, r-1]\!]$, multiplying by b_j, we get, for $t \in [-1,1]$,

$$b_j(t)\int_{t_0}^{t} T_j(s)T_i(s)\mathrm{d}s = -\gamma_{iji}(t_0)\sum_{-d_j\leqslant k\leqslant d_j} b_{j,k}T_k(t) + \sum_{-d_j\leqslant k\leqslant d_j} b_{j,k}\gamma_{ij(i+k)}(t), \tag{4.8}$$

where the second sum follows from $\gamma_{iji}(t)T_k(t) = (\gamma_{ij(i+k)}(t) + \gamma_{ij(i-k)}(t))/2$ and the fact that $b_{j,k} = b_{j,-k}$.

This expression shows that the matrix representation $(\hat{B}_{j,ki})_{k,i\in\mathbb{N}}$ of the Y^1-endomorphism $\varphi \mapsto b_j(t)\int_{t_0}^{t} T_j(s)\varphi(s)\mathrm{d}s$ is sparse and has a so-called *almost-banded structure*. More precisely, it is made of a *horizontal band* of non-zero coefficients $\hat{B}_{j,ki}$, with $0 \leqslant k \leqslant d_j$, $i \in \mathbb{N}$, which we call *initial coefficients*, together with a *diagonal band* of non-zero coefficients $\hat{B}_{j,ki}$, with $i \in \mathbb{N}$ and $i - j - 1 - d_j \leqslant k \leqslant i + j + 1 + d_j$, which we call *diagonal coefficients*. A graphic view of this structure is shown in **Figure 4.1a**.

The following definition formally establishes the notion of almost-banded matrix, in the finite as well as in the infinite case. It is robust in the sense that if an infinite matrix representing an endomorphism of Y^1 is (h,d) almost-banded, then so are all its finite-dimensional truncations (defined in **Section 4.1.3**).

Definition 4.9 *Let $\mathcal{I} = \mathbb{N}$ or $[\![0, n-1]\!]$ (for some $n > 0$) be a set of indices, and h, d two nonnegative integers.*

1. *For $i \in \mathcal{I}$, $v \in \mathbb{R}^{\mathcal{I}}$ is said to be (h,d) almost-banded around index i if for all $j \in \mathcal{I}$, $v_j = 0$ whenever $j > h$ and $|i-j| > d$.*
2. *The square matrix $A = (a_{ij})_{i,j\in\mathcal{I}} \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ is said to be (h,d) almost-banded if for all $j \in \mathcal{I}$, the jth column $v^{(j)} = (a_{ij})_{i\in\mathcal{I}} \in \mathbb{R}^{\mathcal{I}}$ of A is almost-banded around index j.*

It turns out that the matrix representation of $\mathbf{K}$ has an almost-banded structure: to obtain $\mathbf{K}\cdot T_i$, it suffices to sum all the contributions from Equation (4.8) for $0 \leqslant j < r$. Hence $\mathbf{K}\cdot T_i$ is (h,d) almost-banded around index i, which shows that the integral operator $\mathbf{K}$ has an (h,d) almost-banded matrix representation, where:

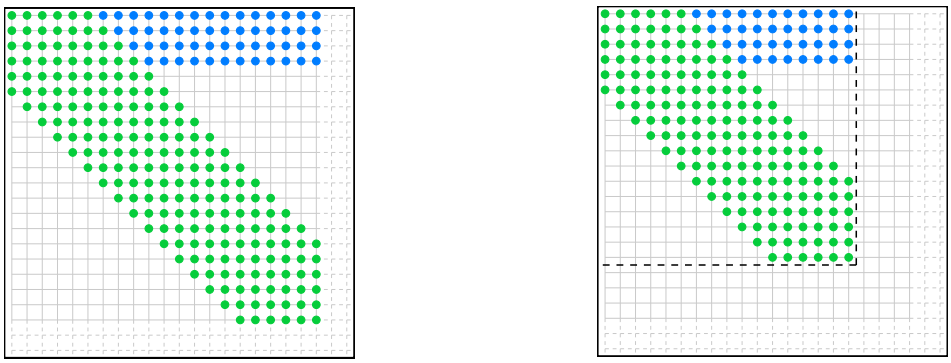
$$h = \max_{0\leqslant j<r} d_j, \qquad d = \max_{0\leqslant j<r} j + 1 + d_j.$$

The width of the horizontal band is $h+1$ and that of the diagonal band is $2d+1$. With a slight terminology abuse, such operators are directly called *almost-banded operators* in what follows.

Iterations of K in Y^1 and almost-banded approximations of $(1+\mathbf{K})^{-1}$

We recalled in **Section 4.1.1** the convergence of the Neumann series $\mathbf{1} - \mathbf{K} + \mathbf{K}^2 - \dots$ to $(\mathbf{1}+\mathbf{K})^{-1}$ in $(\mathcal{C}^0, \|\cdot\|_\infty)$. The following lemma establishes an analogous result in Y^1:

Lemma 4.10 *The operator $\mathbf{1}+\mathbf{K}$ is invertible in Y^1 and its inverse is given by the Neumann series $\sum_{i\geqslant 0}(-\mathbf{K})^i$ which converges in $o(\varepsilon^n)$ for all $\varepsilon > 0$. More precisely:*



(a) Almost-banded structure of operator $\mathbf{K}$.

(b) Truncated operator $\mathbf{K}^{[n]}$.

■ **Figure 4.1**: Almost-banded structure of operator $\mathbf{K}$ and its truncations $\mathbf{K}^{[n]}$.

- $\sum_{i=0}^{n}(-1)^i\mathbf{K}^i$ *is a sequence of* (dn, dn) *almost-banded operators;*
- $\|\sum_{i>n}(-1)^i\mathbf{K}^i\|_{\mathcal{Y}^1} \leqslant \sum_{i>n}(6di+1)\frac{(2C)^i}{i!}$ *(*C *defined in Equation (4.4)).*

Proof. In $\mathcal{Y}^1$, since $\mathbf{K}$ is (h, d) almost-banded, with $h < d$, a straightforward induction shows that $\mathbf{K}^n$ is (h_n, d_n) almost-banded, with $d_n = nd$ and $h_n < d_n$.

Fix an index $j \in \mathbb{N}$. Then the *symmetric* Chebyshev series of $\mathbf{K}^n \cdot T_j$ has at most $(2d_n + 1) + (2h_n + 1) + (2d_n + 1) \leqslant 6nd + 1$ non-zero coefficients. Moreover, each of these (two-sided) coefficients is bounded by $\|\mathbf{K}^n \cdot T_j\|_\infty \leqslant (2C)^n/n!$. Hence, we get:

$$\|\mathbf{K}^n \cdot T_j\|_{\mathcal{Y}^1} \leqslant (6dn + 1)\frac{(2C)^n}{n!},$$

from which we conclude using **Lemma 2.41**. □

This shows that obtaining an approximate solution of (4.3) via iterations of $\mathbf{K}$, is possible both in $(\mathcal{C}^0([-1, 1]), \|\cdot\|_\infty)$ and in $(\mathcal{Y}^1, \|\cdot\|_{\mathcal{Y}^1})$. However, the action of $\mathbf{K}$ and its iterates involves handling an infinite dimensional space. In the sequel, we prove that suitable *truncations* of $\mathbf{K}$ allow for obtaining approximate solutions in finite dimensional subspaces of $\mathcal{Y}^1$ and these solutions converge in $o(\varepsilon^n)$ for all $\varepsilon > 0$ to the exact solution of (4.3).

4.1.3 ▶ Approximate solutions via truncations $\mathbf{K}^{[n]}$ of $\mathbf{K}$

The nth truncation (also called the nth section in [93]) of the integral operator $\mathbf{K}$ is defined as follows:

$$\mathbf{K}^{[n]} := \boldsymbol{\pi}_n \cdot \mathbf{K} \cdot \boldsymbol{\pi}_n. \tag{4.9}$$

The truncation method (also called projection method in [93]) to solve Equation (4.3) consists in replacing $\mathbf{K}$ by $\mathbf{K}^{[n]}$ and solving the finite-dimensional linear problem:

$$\varphi + \mathbf{K}^{[n]} \cdot \varphi = \psi. \tag{4.10}$$

Note that the order $n+1$ square matrix M representing $\mathbf{K}^{[n]}$ is the initial $n+1$ square submatrix of the infinite dimensional matrix representation of $\mathbf{K}$. Therefore, M has an (h,d) almost-banded structure (see **Figure 4.1b**). This implies that solving Equation (4.10) reduces to solving a linear system of equations with a specific almost-banded structure. We revisit in **Section 4.2** efficient algorithms for solving such systems.

Moreover, we prove the following important fast convergence result:

Theorem 4.11 *Let $\varphi^* := (\mathbf{I}+\mathbf{K})^{-1}\cdot\psi$ be the exact solution of integral equation (4.3) and $\varphi_n^\circ := (\mathbf{I}+\mathbf{K}^{[n]})^{-1}\cdot\psi$ be the solution of the truncated system (4.10). We have:*

$$\|\varphi^* - \varphi_n^\circ\|_{\mathcal{Y}^1} = o(\varepsilon^n) \quad \textit{for all } \varepsilon > 0.$$

In **[19,** Thm. 4.4**]** and **[191,** Thm. 4.5**]**, similar convergence rates were proven in the different context of the uniform norm and for rather different approximations schemes: either the considered operator is different (the differential operator is handled in **[191]**) or the employed tools are more involved (*main asymptotic existence theorem* for linear recurrences is needed in **[19]**). The proof of **Theorem 4.11** requires important theoretical properties concerning the truncated operator $\mathbf{K}^{[n]}$ in relation with $\mathbf{K}$ in the space $\mathcal{Y}^1$, which are given in the next two additional lemmas. They are also of first importance for the validation method developed in **Section 4.3**.

Firstly, let us prove that $\mathbf{K}^{[n]}$ is a good approximation of $\mathbf{K}$ for the $\mathcal{Y}^1$ norm.

Lemma 4.12 *Let $\mathbf{K}$ be the integral operator in (4.3), of order r and polynomial coefficients b_j. Let (h,d) be the parameters of its almost-banded structure and $n \geqslant r+d$ be the truncation order, then:*

(4.12 i) $\mathbf{K}^{[n]}\cdot T_i = \mathbf{K}\cdot T_i$ *for all* $i \leqslant n-d$.

(4.12 ii) $\mathbf{K}^{[n]} \to \mathbf{K}$ *in* $\mathcal{Y}^1$ *as* $n \to +\infty$. *More precisely:*

$$\|\mathbf{K}-\mathbf{K}^{[n]}\|_{\mathcal{Y}^1} \leqslant B\max\left(\frac{1}{n+1-r-d}, \frac{2}{n-r}\right) \quad \textit{with } B = \sum_{j=0}^{r-1}\|b_j\|_{\mathcal{Y}^1},$$

which implies a convergence speed of $\mathcal{O}(1/n)$ as $n \to +\infty$.

Proof. For (4.12 i), if $i \leqslant n-d$, then $\mathbf{K}\cdot T_i$ is of degree at most $\max(h, n-d+d) = n$ because $n \geqslant d \geqslant h$. Hence:

$$\mathbf{K}^{[n]}\cdot T_i := \boldsymbol{\pi}_n\cdot\mathbf{K}\cdot\boldsymbol{\pi}_n\cdot T_i = \boldsymbol{\pi}_n\cdot\mathbf{K}\cdot T_i = \mathbf{K}\cdot T_i.$$

For (4.12 ii), note first that from **Lemmas 2.41** and (4.12 i), one has:

$$\|\mathbf{K}-\mathbf{K}^{[n]}\|_{\mathcal{Y}^1} = \sup_{i\geqslant 0}\|\mathbf{K}\cdot T_i - \mathbf{K}^{[n]}\cdot T_i\|_{\mathcal{Y}^1} = \sup_{i>n-d}\|\mathbf{K}\cdot T_i - \mathbf{K}^{[n]}\cdot T_i\|_{\mathcal{Y}^1}.$$

Now, for $\varphi \in \mathcal{Y}^1$ one has the following decomposition:

$$(\mathbf{K}-\mathbf{K}^{[n]})\varphi = \mathbf{K}\cdot\varphi - \boldsymbol{\pi}_n\cdot\mathbf{K}\cdot\boldsymbol{\pi}_n\cdot\varphi = \mathbf{K}\cdot(\mathbf{1}-\boldsymbol{\pi}_n)\cdot\varphi + (\mathbf{1}-\boldsymbol{\pi}_n)\cdot\mathbf{K}\cdot\boldsymbol{\pi}_n\cdot\varphi.$$

Hence, one can evaluate $\mathbf{K}-\mathbf{K}^{[n]}$ on all remaining T_i's for $i > n-d$:

- If $n-d<i\leqslant n$, then $(\mathbf{K}-\mathbf{K}^{[n]})\cdot T_i=(\mathbf{1}-\mathbf{\Pi}_n)\cdot\mathbf{K}\cdot T_i$. Note that, since $n\geqslant h$, only the diagonal coefficients of $\mathbf{K}\cdot T_i$ may bring a nonzero contribution. Moreover, we have $i\pm j\pm 1\geqslant n+1-d-r$. From that we deduce an upper bound of the approximation error:
$$\|(\mathbf{1}-\mathbf{\Pi}_n)\cdot\mathbf{K}\cdot T_i\|_{\mathrm{L}^1}\leqslant\frac{B}{n+1-r-d}.$$
- If $i>n$, then $(\mathbf{K}-\mathbf{K}^{[n]})\cdot T_i=\mathbf{K}\cdot T_i$. We have that $i\pm j\pm 1\geqslant n-r$ for $0\leqslant j<r$. Hence:
$$\|\mathbf{K}\cdot T_i\|_{\mathrm{L}^1}\leqslant\frac{2B}{n-r}$$

We conclude by taking the maximum of these two bounds. □

The convergence of $\mathbf{K}^{[n]}$ to $\mathbf{K}$ also implies that $\mathbf{1}+\mathbf{K}^{[n]}$ is invertible for n large enough:

Lemma 4.13 *For n large enough, we have:*

(4.13 i) *the endomorphism $\mathbf{1}+\mathbf{K}^{[n]}$ is invertible.*

(4.13 ii) *$(\mathbf{1}+\mathbf{K}^{[n]})^{-1}$ converges to $(\mathbf{1}+\mathbf{K})^{-1}$, with:*
$$\begin{aligned}\|(\mathbf{1}+\mathbf{K}^{[n]})^{-1}-(\mathbf{1}+\mathbf{K})^{-1}\|_{\mathrm{L}^1}&\leqslant\frac{\|(\mathbf{1}+\mathbf{K})^{-1}\|_{\mathrm{L}^1}^2}{1-\|(\mathbf{1}+\mathbf{K})^{-1}\cdot(\mathbf{K}-\mathbf{K}^{[n]})\|_{\mathrm{L}^1}}\|\mathbf{K}-\mathbf{K}^{[n]}\|_{\mathrm{L}^1}\\&=\mathcal{O}\left(\frac{1}{n}\right)\quad\text{as } n\to+\infty\end{aligned}$$

(4.13 iii) $(\mathbf{1}+\mathbf{K}^{[n]})^{-1}=\sum_{i\geqslant 0}(-\mathbf{K}^{[n]})^i$.

Proof. For (4.13 i) and (4.13 ii), using the bound in $\mathcal{O}(1/n)$ for $\|\mathbf{K}-\mathbf{K}^{[n]}\|_{\mathrm{L}^1}$ obtained in Lemma 4.12, the invertibility of $\mathbf{1}+\mathbf{K}^{[n]}$ as well as the announced explicit upper bound for $\|(\mathbf{1}+\mathbf{K}^{[n]})^{-1}-(\mathbf{1}+\mathbf{K})^{-1}\|_{\mathrm{L}^1}$ directly follow from [93, Chap. 2, Cor. 8.2].

For (4.13 iii), since by Lemma 4.10 the Neumann series of $\mathbf{K}$ absolutely converges, there is a $p>0$ such that $\|\mathbf{K}^p\|_{\mathrm{L}^1}<1$. Since $\mathbf{K}^{[n]}\to\mathbf{K}$ as $n\to+\infty$, there is an n such that $\|(\mathbf{K}^{[n]})^p\|_{\mathrm{L}^1}<1$. Therefore, the Neumann series of $(\mathbf{K}^{[n]})^p$ is absolutely convergent, and the following factorization establishes the absolute convergence of the Neumann series of $\mathbf{K}^{[n]}$:
$$\sum_{i\geqslant 0}(-\mathbf{K}^{[n]})^i=\left(\sum_{i<p}(-\mathbf{K}^{[n]})^i\right)\cdot\left(\sum_{i\geqslant 0}(-\mathbf{K}^{[n]})^{pi}\right)$$
□

Note that from the previous lemma, one readily obtains that $\varphi_n^\circ:=(\mathbf{1}+\mathbf{K}^{[n]})^{-1}\cdot\psi$ converges to the exact solution $\varphi^*:=(\mathbf{1}+\mathbf{K})^{-1}\cdot\psi$ in $\mathcal{O}(1/n)$. However, we can now prove the far better convergence result of the main Theorem 4.11.

Proof of Theorem 4.11. Take $n>d$ large enough so that $\mathbf{1}+\mathbf{K}^{[n]}$ is invertible by Lemma 4.13. Let $\overline{\varphi}_n=(\sum_{i\leqslant\lfloor n/2d\rfloor}(-1)^i\mathbf{K}^i)\cdot\psi$ denote the approximate solution obtained by computing the Neumann series of $\mathbf{K}$ at order $\lfloor n/2d\rfloor$. Since this series is an $(d\lfloor n/2d\rfloor,d\lfloor n/2d\rfloor)$ almost-banded

operator, we get that $\overline{\varphi}_n$ is a polynomial of degree at most $\deg(\psi) + d\lfloor n/2d \rfloor \leqslant \deg(\psi) + n/2$. Hence, for n large enough, the degree of $\overline{\varphi}_n$ does not exceed $n - d$, so that we have the key equality $\mathbf{K}^{[n]} \cdot \overline{\varphi}_n = \mathbf{K} \cdot \overline{\varphi}_n$, according to **Lemma (4.12 i)**. From that we deduce:

$$\varphi^* - \varphi_n^\circ = \left(\mathbf{1} - \left(\mathbf{1} + \mathbf{K}^{[n]}\right)^{-1} (\mathbf{1} + \mathbf{K})\right) \cdot (\varphi^* - \overline{\varphi}_n)\,.$$

From **Lemma (4.13 ii)** and **Lemma 4.10**, we finally get:

$$\|\varphi^* - \varphi_n^\circ\|_{\curlyvee^1} = \mathcal{O}\left(\frac{(2C)^{\lfloor n/2d \rfloor}}{\lfloor n/2d \rfloor !}\right),$$

which is an $o(\varepsilon^n)$ for all $\varepsilon > 0$. □

For completeness, we note the following alternative proof of **Lemma 4.10**. The convergence of the finite-dimensional truncations $\mathbf{K}^{[n]}$ to $\mathbf{K}$ in $\curlyvee^1$ implies that $\mathbf{K}$ is a compact endomorphism of the Banach space $\curlyvee^1$. The Fredholm alternative **[41]** says in that case that $\mathbf{1} + \mathbf{K} : \curlyvee^1 \to \curlyvee^1$ is injective if and only if it is surjective. But, as mentioned at the beginning of **Section 4.1.2**, we already have the injectivity of this operator. Hence, we conclude that $\mathbf{1} + \mathbf{K}$ is bijective, and moreover that it is a bicontinuous isomorphism of $\curlyvee^1$ (using the Banach continuous inverse theorem).

We discuss in the next section algorithms concerning almost-banded matrices, since this structure is essential both for efficient algorithmic computation of φ° and its *a posteriori* validation step.

4.2 Algorithms involving almost-banded matrices

Let A and B be two order n square matrices, respectively (h_A, d_A) and (h_B, d_B) almost-banded. In **Table 4.1** we recall several elementary operations which are straightforward, the result is an almost-banded matrix, and their complexity is in $\mathcal{O}(n)$ provided that the almost-banded parameters are supposed constant with respect to n.

Note that the product AB is computable in $\mathcal{O}(n(h_A + d_A)(h_B + d_B))$ operations by applying the evaluation on a sparse vector (line 5 in the table) on all the columns of B. In the dense case, when $h_A + d_A \approx n$ and $h_B + d_B \approx n$, this corresponds to the naive $\mathcal{O}(n^3)$ algorithm, hence a fast multiplication algorithm may become more appropriate.

In ChebValid, we provide a type `mpfi_bandvec_t` for almost-banded vectors, another type `mpfi_bandmatrix_t` for almost-banded matrices, and operations on the latter for addition, negation, subtraction, evaluation on an almost-banded vector, multiplication and 1-norm. Note that corresponding operations are also available for `double` and `mpfr_t` coefficient types.

We now turn to efficient algorithms for solving almost-banded linear systems as well as matrix inversion. In **Section 4.2.1**, we recall Olver and Townsend's algorithm for solving order n almost-banded linear systems in linear complexity with respect to n. This directly leads to a

Operation	Result's a.-b. structure	Complexity
λA, with $\lambda \in \mathbb{R}$	(h_A, d_A)	$\mathcal{O}(n(h_A + d_A))$
$A + B$ or $A - B$	$(\max(h_A, h_B), \max(d_A, d_B))$	$\mathcal{O}(n(\max(h_A, h_B) + \max(d_A, d_B)))$
Av dense $v \in \mathbb{R}^n$	dense	$\mathcal{O}(n(h_A + d_A))$
Av (h_v, d_v) a.-b.$v \in \mathbb{R}^n$	$(\max(h_v + d_A, h_A), d_v + d_A)$	$\mathcal{O}((h_A + d_A)(h_v + d_v))$
AB	$(\max(h_B + d_A, h_A), d_B + d_A)$	$\mathcal{O}(n(h_A + d_A)(h_B + d_B))$
$\|A\|_1$	-	$\mathcal{O}(n(h_A + d_A))$

■ **Table 4.1**: Elementary operations on almost-banded (a.-b.) matrices or vectors: A and B are order n square matrices, respectively (h_A, d_A) and (h_B, d_B) almost-banded, and $v \in \mathbb{R}^n$ is either dense or almost-banded around some index $i \in [\![0, n-1]\!]$.

quadratic algorithm for inverting an almost-banded matrix. To achieve linear complexity for inversion, we give in **Section 4.2.2** a modified version of this algorithm.

4.2.1 ▶ A reminder on Olver and Townsend's algorithm for almost-banded linear systems

Let M denote an (h, d) almost-banded order n square matrix with $h \leqslant d$, and $y \in \mathbb{R}^n$. The goal is to solve an almost-banded linear system $Mx = y$ for unknown $x \in \mathbb{R}^n$. The procedure is split into two parts. First, a QR decomposition $QM = R$ is computed, with Q orthogonal and R upper triangular. Then, the equivalent system $Rx = Qy$ is solved by back-substitution. The key challenge is to maintain a linear complexity with respect to n in both steps.

■ First step: QR decomposition

This is computed in **Algorithm** OlverTownsendQR using Givens rotations' method which eliminates line after line the coefficients of M under the diagonal to finally obtain R, as shown in **Figure 4.2**.

More precisely, at step i, for each $j \in [\![i+1, \min(i+d, n-1)]\!]$, we apply a well-chosen rotation $\begin{pmatrix} c_{ij} & -s_{ij} \\ s_{ij} & c_{ij} \end{pmatrix}$ on lines i and j in order to get $R_{ji} = 0$. Note that at the end of each step i, $R_{ii} \neq 0$ if and only if the matrix M is invertible.

The direct application of this process would cause the progressive filling-in of the rows, which would give a dense upper triangular matrix R. In fact, this phenomenon can be controlled by noticing that for each $i < n - 2d - 1$, the "end of the row" i of R, $(R_{i,i+2d+1}, \ldots, R_{i,n-1})$, is a linear combination of the corresponding dense part of M: $(M_{\ell,i+2d+1}, \ldots, M_{\ell,n-1})$ for $\ell \in [\![0, h]\!]$. Hence, it suffices to manipulate instead the coefficients $\lambda_{i\ell}$ of the linear combination:

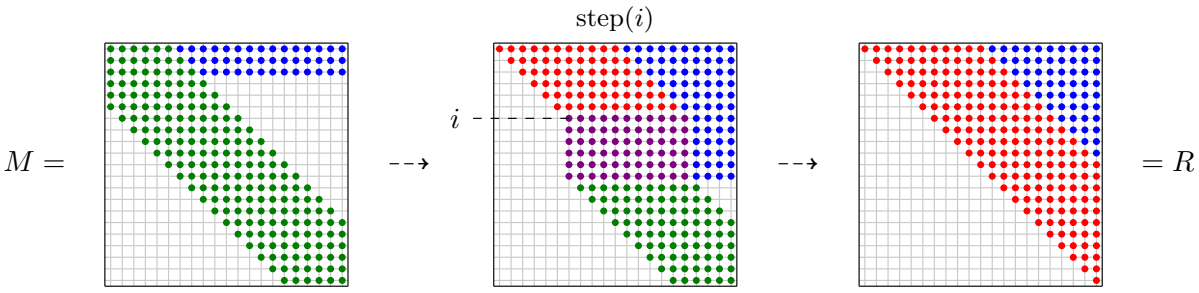


Figure 4.2: Step 1 (OlverTownsendQR) of Olver and Townsend's algorithm.

$$(R_{i,i+2d+1}, \ldots, R_{i,n-1}) = \sum_{\ell=0}^{h} \lambda_{i\ell} (M_{\ell,i+2d+1}, \ldots, M_{\ell,n-1}). \tag{4.11}$$

Based on this observation, **Algorithm** OlverTownsendQR(M) returns the QR decomposition $QM = R$ under the following representation:

- Q is completely determined by c_{ij}, s_{ij}:

$$Q = \prod_{i=0}^{n-1} \prod_{j=i+1}^{\min(i+d,n-1)} Q^{(ij)},$$

  where the $Q^{(ij)}$ are rotation matrices defined by:

$$(Q^{(ij)})_{k\ell} = \begin{cases} 1 & \text{if } k = \ell \text{ and } k \neq i, j, \\ c_{ij} & \text{if } k = \ell = i \text{ or } k = \ell = j, \\ s_{ij} & \text{if } k = j \text{ and } \ell = i, \\ -s_{ij} & \text{if } k = i \text{ and } \ell = j, \\ 0 & \text{otherwise.} \end{cases} \tag{4.12}$$

- R is upper triangular and represented by its $2d+1$ upper diagonals (entries R_{ij} for $i \in [\![0, n-1]\!]$ and $j \in [\![i, \min(i+2d, n-1)]\!]$ are given explicitly) together with the coefficients $\lambda_{i\ell}$ ($i \in [\![0, n-1]\!]$ and $\ell \in [\![0, h]\!]$) defining the rest of R as in (4.11).

Formally, one has:

Proposition 4.14 *Algorithm* OlverTownsendQR *applied on an (h, d) almost-banded matrix of order n with $h \leqslant d$ is correct and runs in $\mathcal{O}(nd^2)$ operations.*

Proof. Given in [191]. □

In ChebValid, **Algorithm** OlverTownsendQR is only implemented on floating-point matrices, not interval ones, due to the remarks about interval matrix inversion in **Section 1.3**. We provide a type `mpfr_bandmatrix_QRdecomp_t` for the decomposition (Q, R) of an almost-banded matrix. It is computed by the function `mpfr_bandmatrix_get_QRdecomp`, which implements **Algorithm** OlverTownsendQR.

Algorithm 4.2 OlverTownsendQR(M) – Step 1 of Olver and Townsend's algorithm

Input: an order n, (h,d) almost-banded matrix M with $h \leqslant d$.
Output: a QR factorization $QM = R$: Q defined by c_{ij}, s_{ij} as in (4.12); R defined by R_{ij} $(i \in [\![0, n-1]\!], j \in [\![i, \min(i+2d, n-1)]\!])$ and $\lambda_{i\ell}$ as in (4.11).

$R \leftarrow M$
for $i = 0$ **to** $n-1$ and $j = 0$ **to** h **do** $\lambda_{ij} \leftarrow 0$
for $i = 0$ **to** h **do** $\lambda_{ii} \leftarrow 1$
for $i = 0$ **to** $n-1$ **do**
  for $j = i+1$ **to** $\min(i+d, n-1)$ **do**
    if $R_{ji} = 0$ **then**
      $c_{ij} \leftarrow 1$ and $s_{ij} \leftarrow 0$
    else
      $r \leftarrow \sqrt{R_{ii}^2 + R_{ij}^2}$
      $c_{ij} \leftarrow R_{ii}/r$ and $s_{ij} \leftarrow -R_{ji}/r$
      for $k = i$ **to** $\min(i+2d, n-1)$ **do** $\begin{pmatrix} R_{ik} \\ R_{jk} \end{pmatrix} \leftarrow \begin{pmatrix} c_{ij} & -s_{ij} \\ s_{ij} & c_{ij} \end{pmatrix} \begin{pmatrix} R_{ik} \\ R_{jk} \end{pmatrix}$
      for $\ell = 0$ **to** h **do** $\begin{pmatrix} \lambda_{i\ell} \\ \lambda_{j\ell} \end{pmatrix} \leftarrow \begin{pmatrix} c_{ij} & -s_{ij} \\ s_{ij} & c_{ij} \end{pmatrix} \begin{pmatrix} \lambda_{i\ell} \\ \lambda_{j\ell} \end{pmatrix}$
    end if
  end for
end for

Second step: back-substitution

Once step 1 is performed and returns $QM = R$, we first apply the rotations $Q^{(ij)}$ on the right hand side $y \in \mathbb{R}^n$ to obtain Qy in $\mathcal{O}(nd)$ operations. Now we have to solve $Rx = Qy := y_Q$.

If R is regarded as a dense upper triangular matrix, the classical back-substitution algorithm requires $\mathcal{O}(n^2)$ operations. However, based on the sparse representation of R, the back-substitution in **Algorithm** OlverTownsendBackSubs is more efficient. Its main idea to compute the solution x_i (for i going backwards from $n-1$ to 0) is to use Equation (4.11) for expressing R_{ij} as soon as $i < n-2d-1$, $j > i$:

$$x_i = \left(y_{Qi} - \sum_{j=i+1}^{n-1} R_{ij} x_j \right) / R_{ii} = \left(y_{Qi} - \sum_{j=i+1}^{i+2d} R_{ij} x_j - \sum_{\ell=0}^{h} \lambda_{i\ell} z_{i\ell} \right) / R_{ii},$$

where

$$z_{i\ell} = (M_{\ell,i+2d+1}, \dots, M_{\ell,n-1})(x_{i+2d+1}, \dots, x_{n-1})^T = \sum_{j=i+2d+1}^{n-1} M_{\ell j} x_j.$$

Then, once $z_{i\ell}$ is computed, $z_{i-1,\ell}$ is updated in constant time:

$$z_{i-1,\ell} = M_{\ell,i+2d} x_{i+2d} + z_{i\ell}.$$

This leads to the following proposition:

Proposition 4.15 *Algorithm* OlverTownsendBackSubs *is correct and requires* $\mathcal{O}(nd)$ *operations.*

Proof. Given in [191]. □

In ChebValid, **Algorithm** OlverTownsendBackSubs is implemented by the function `mpfr_bandmatrix_QRdecomp_solve_fr`

Algorithm 4.3 OlverTownsendBackSubs$(M, (Q, R), y)$ – Step 2 of O. & T.'s algorithm

Input: an order n invertible (h, d) almost-banded M with $h \leqslant d$, its QR decomposition (Q, R) produced by OlverTownsendQR(M) and a vector $y \in \mathbb{R}^n$.
Output: the solution vector x of $Mx = y$.

▷ *Compute* Qy
for $i = 0$ **to** $n - 1$ **do**
  for $j = i + 1$ **to** $\min(i + d, n - 1)$ **do**
    $\begin{pmatrix} y_i \\ y_j \end{pmatrix} \leftarrow \begin{pmatrix} c_{ij} & -s_{ij} \\ s_{ij} & c_{ij} \end{pmatrix} \begin{pmatrix} y_i \\ y_j \end{pmatrix}$
  end for
end for

▷ *Back-substitution*
for $\ell = 0$ **to** h **do** $z_\ell \leftarrow 0$
for $i = n - 1$ **down to** 0 **do**
  ▷▷ *Update* z_ℓ
  if $i + 2d + 1 < n$ **then for** $\ell = 0$ **to** h **do** $z_\ell \leftarrow z_\ell + M_{\ell,i+2d+1} x_{i+2d+1}$
  ▷▷ *Compute* x_i
  $x_i \leftarrow \left(y_i - \sum_{j=i+1}^{\min(i+2d,n-1)} R_{ij} x_j - \sum_{\ell=0}^{h} \lambda_{i\ell} z_\ell \right) / R_{ii}$
end for

4.2.2 ▶ An algorithm for almost-banded approximation of inverse of almost-banded matrix

Based on Olver and Townsend's algorithm, the inverse of an (h, d) almost-banded order n matrix M (with $h \leqslant d$) can be computed in quadratic time $\mathcal{O}(n^2 d)$. First, OlverTownsendQR(M) is performed in $\mathcal{O}(nd^2)$ operations (**Proposition 4.14**) to obtain a QR decomposition $QM = R$. Then, each column $v^{(i)}$ of index $i \in [\![0, n-1]\!]$ of M^{-1} is computed by solving $Mv^{(i)} = e^{(i)}$, where $e^{(i)}$ denotes the ith vector of the canonical basis of $\mathbb{R}^n$. This is achieved by using n times step 2, resulting in a total of $\mathcal{O}(n^2 d)$ operations.

Unfortunately, this algorithm has quadratic running time and returns a dense inverse matrix representation. In some cases however, such as the validation process developed in **Section 4.3**, a *sparse approximation* of M^{-1} is sufficient. As proved in **Lemma 4.13** (iii), the inverse of $M = \mathbf{1} + \mathbf{K}^{[n]}$ can be approximated with almost-banded matrices. This leads to adapting the full inversion procedure described above to compute only coefficients on diagonal and horizontal bands.

Let $A \simeq M^{-1}$ be the required approximate inverse with an almost-banded structure given by the parameters (h', d') (we do not require $h' \leqslant d'$). Firstly, one computes the QR decomposition $QM = R$ in $\mathcal{O}(nd^2)$ operations using OlverTownsendQR(M). Then, Step 2 of Olver and Townsend's algorithm is modified, resulting into **Algorithm** SparseBackSubs. For each $i \in [\![0, n-1]\!]$, the ith column $v^{(i)}$ of A is computed as an approximate solution of $Rx = Qe^{(i)}$, in the form of an (h', d') almost-banded vector around index i:

1. $Qe^{(j)} \in \mathbb{R}^n$ is computed only partially, between entries $i - d$ and $i + d'$. Note that in general $Qe^{(j)}$ has zero entries between indices 0 and $i - d - 1$, and is dense from $i - d$ to $n - 1$.

2. The back-substitution only computes entries of the solution from indices $i + d'$ to $i - d'$, and from h' to 0. Since the other entries are implicitly set to 0, these computed coefficients are only *approximations* of the entries at the same position in the exact solution. But considering that the neglected entries were small enough, this approximation is expected to be convenient.

We provide a complexity analysis of **Algorithm** SparseBackSubs, but nothing is stated concerning the accuracy of the obtained approximation. This procedure should really be seen as a heuristic in general.

Proposition 4.16 ***Algorithm*** SparseBackSubs *involves* $\mathcal{O}((h + d)(h' + d'))$ *operations.*

Proof. The first step (computing the diagonal coefficients of Qy) clearly requires $\mathcal{O}(dd')$ arithmetic operations. Now consider the second step (the partial back-substitution) and enter the main loop at line 11, where index j lives in a set of size $\mathcal{O}(h' + d')$. First, we need to update the values z_ℓ. At first sight, each z_ℓ seems to involve a sum of $\mathcal{O}(h' + d')$ terms. But in fact, the total amortized cost related to line 13 is $\mathcal{O}((h' + d')h)$, since at the end of the algorithm, each z_ℓ is equal to $\sum_{k \in [\![2d+1, n-1]\!] \cap (\mathfrak{D} \cup \mathfrak{H})} M_{\ell k} x_k$, which is a sum of $\mathcal{O}(h' + d')$ terms. As a matter of fact, $j_z \leqslant j + 2d + 1$ most of the time, except when $h' < i - d'$ and the current index j falls from $i - d'$ to h'. After that, the computation of x_j involves two sums with a total of $\mathcal{O}(h + d)$ terms. We therefore obtain the claimed complexity. □

Corollary 4.17 ***Algorithm*** AlmostBandedApproxInverse$(M, (Q, R), h', d')$ *produces an* (h', d') *almost-banded approximation of the inverse of an order* n*,* (h, d) *almost-banded matrix* M *in* $\mathcal{O}(n(h + d)(h' + d'))$ *operations.*

In ChebValid, **Algorithm** AlmostBandedApproxInverse is implemented by the function `mpfr_bandmatrix_QRdecomp_approx_band_inverse`.

We now turn to the *a posteriori* validation step.

4.3 A quasi-Newton validation method

Given an approximate solution φ° of the integral equation (4.3), we propose an *a posteriori* validation method which computes a rigorous upper bound for the approximation error $\|\varphi^* -$

Algorithm 4.4 SparseBackSubs$(M, (Q, R), h', d', i)$ – A.-b. approximate column inversion

Input: an order n, (h, d) almost-banded matrix M with the QR decomposition $QM = R$ produced by OlverTownsendQR(M) and parameters h', d', i with $h' \in [\![h, n-1]\!]$, $d' \in [\![d, n-1]\!]$ and $i \in [\![0, n-1]\!]$.

Output: (h', d') almost-banded vector x around index i such that $Mx \approx e^{(i)}$.

$\mathfrak{D} \leftarrow [\![i - d', i + d']\!] \cap [\![0, n-1]\!]$ and $\mathfrak{H} \leftarrow [\![0, h']\!] - \mathfrak{D}$

▷ *Compute diagonal coefficients of Qy*

for j **in** $\mathfrak{D} \cup [\![i + d' + 1, i + d' + d]\!] - \{i\}$ **do** $y_j \leftarrow 0$

$y_i \leftarrow 1$

for j **in** $\mathfrak{D}$ going upwards **do**

  for k **in** $[\![j + 1, j + d]\!] \cap [\![0, n-1]\!]$ going upwards **do**

$$\begin{pmatrix} y_j \\ y_k \end{pmatrix} \leftarrow \begin{pmatrix} c_{jk} & -s_{jk} \\ s_{jk} & c_{jk} \end{pmatrix} \begin{pmatrix} y_j \\ y_k \end{pmatrix}$$

  end for

end for

▷ *Partial back-substitution*

for $\ell \in [\![0, h]\!]$ **do** $z_\ell \leftarrow 0$

$j_z \leftarrow n - 1$

for j **in** $\mathfrak{D} \cup \mathfrak{H}$ going downwards **do**

  ▷▷ *Update z_ℓ*

  if $j + 2d < j_z$ **then**

    for $\ell \in [\![0, h]\!]$ **do** $z_\ell \leftarrow z_\ell + \sum_{k \in [\![j+2d+1, j_z]\!] \cap (\mathfrak{D} \cup \mathfrak{H})} M_{\ell k} x_k$

    $j_z \leftarrow j + 2d$

  end if

  ▷▷ *Compute x_j*

  if $j \in \mathfrak{D}$ **then** $c \leftarrow y_j$ **else** $c \leftarrow 0$

$$x_j \leftarrow \left(c - \sum_{k \in [\![j+1, j+2d]\!] \cap (\mathfrak{D} \cup \mathfrak{H})} R_{jk} x_k - \sum_{\ell=0}^{h} \lambda_{j\ell} z_\ell \right) / R_{jj}$$

end for

Algorithm 4.5 AlmostBandedApproxInverse$(M, (Q, R), h', d')$ – A.-b. approximate inverse

Input: an order n, (h, d) almost-banded matrix M with the QR decomposition $QM = R$ produced by OlverTownsendQR(M) and parameters h', d' with $h' \in [\![h, n-1]\!]$ and $d' \in [\![d, n-1]\!]$.

Output: an (h', d') almost-banded matrix A with $A \approx M^{-1}$.

for $i = 0$ **to** $n - 1$ **do**

  $W \leftarrow$ SparseBackSubs$(M, (Q, R), h', d', i)$, approximating $M^{-1} e^{(i)}$

  Set ith column of A to W

end for

$\varphi^\circ\|_{\mathcal{Y}^1}$, where φ^* denotes the exact solution of (4.3). This is based on the general quasi-Newton framework explained in Section 3.3. In this case, $\mathbf{F}\cdot\varphi := \varphi+\mathbf{K}\cdot\varphi-\psi$ is affine, with linear part $\mathbf{1}+\mathbf{K}$. The quasi-Newton method requires an approximate inverse operator $\mathbf{A} \approx (\mathbf{1}+\mathbf{K})^{-1}$ such that $\|\mathbf{1}-\mathbf{A}\cdot(\mathbf{1}+\mathbf{K})\|_{\mathcal{Y}^1} < 1$. Of course, computing an exact inverse would solve the problem but is out of reach. Instead of that, from Lemma 4.13, we know that for a truncation order N_{val} chosen large enough, $(\mathbf{1}+\mathbf{K}^{[N_{\text{val}}]})^{-1}$ exists and is a good approximation of $(\mathbf{1}+\mathbf{K})^{-1}$. Since $(\mathbf{1}+\mathbf{K}^{[N_{\text{val}}]})^{-1}$ is defined by an $(N_{\text{val}}+1)$-order square matrix (its restriction over $\boldsymbol{\pi}_{N_{\text{val}}}\cdot\mathcal{Y}^1$) extended over the whole space $\mathcal{Y}^1$ by the identity, we define the operator $\mathbf{A}$ over $\mathcal{Y}^1$ as an $(N_{\text{val}}+1)$-order square matrix A approximating $(\mathbf{1}+\mathbf{K}^{[N_{\text{val}}]})^{-1}$ over $\boldsymbol{\pi}_{N_{\text{val}}}\cdot\mathcal{Y}^1$, extended by the identity over the whole space:

$$\mathbf{A}\cdot\varphi = A\cdot\boldsymbol{\pi}_{N_{\text{val}}}\cdot\varphi + (\mathbf{1}-\boldsymbol{\pi}_{N_{\text{val}}})\cdot\varphi.$$

The first technical issue is to numerically compute (or represent) both very accurately and efficiently such a matrix A. Specifically, we aim both for a linear complexity computation with respect to N_{val} and for minimizing $\|1_{N_{\text{val}}+1} - A\cdot M\|_1$, where M is an order $N_{\text{val}}+1$ matrix representation for $\mathbf{1}+\mathbf{K}^{[N_{\text{val}}]}$. Among several possibilities to achieve these two requirements, we found none optimal for both. Therefore, we propose two solutions:

S1. As seen in Section 4.2, Olver and Tonwsend's **Algorithm** OlverTownsendBackSubs can be used to numerically compute M^{-1}. The main advantage is that the approximation error $\|1_{N_{\text{val}}+1} - A\cdot M\|_1$ is really close to 0 using standard precision in the underlying computations. Drawback is the quadratic complexity in $\mathcal{O}({N_{\text{val}}}^2 d)$.

S2. Our new heuristic approach is based on **Lemma (4.13 iii)** which states that $(\mathbf{1}+\mathbf{K}^{[N_{\text{val}}]})^{-1}$ is well approximated by almost-banded matrices. So it is natural to look for a matrix A with a (h', d') almost-banded structure. Given h' and d', **Algorithm** AlmostBandedApproxInverse, detailed in Section 4.2, produces an (h', d') almost-banded approximation A of $(\mathbf{1}+\mathbf{K}^{[N_{\text{val}}]})^{-1}$ in $\mathcal{O}(N_{\text{val}}(h'+d')(h+d))$ arithmetic operations (Corollary 4.17). If the parameters (h', d') of the almost-banded structure of A can be chosen small enough compared to N_{val}, this alternative method should be substituted to the standard one.

Deciding which of these two methods should be used in practice is non-trivial: while the second one is more appealing due to the resulting sparsity of A, unfortunately nothing is said about the order of magnitude of N_{val} such that the conclusion of **Lemma (4.13 iii)** is valid, nor about the precise speed of convergence of the Neumann series of M, which would give a good intuition for the values of h' and d' to choose. In what follows, the complexity analysis is thus provided for both cases: a sparse vs. a dense structure of the matrix A. This will allow us to discuss in detail the choice of these parameters in Section 4.3.2.

Next, one has to provide a rigorous Lipschitz constant μ (required by **Theorem 3.9**) for the Newton-like operator. We have:

$$\|\mathbf{1}-\mathbf{A}\cdot(\mathbf{1}+\mathbf{K})\|_{\mathcal{Y}^1} \leqslant \|\mathbf{1}-\mathbf{A}\cdot(\mathbf{1}+\mathbf{K}^{[N_{\text{val}}]})\|_{\mathcal{Y}^1} + \|\mathbf{A}\cdot(\mathbf{K}-\mathbf{K}^{[N_{\text{val}}]})\|_{\mathcal{Y}^1}, \tag{4.13}$$

which can be interpreted as [111]:

- $\|\mathbf{1}-\mathbf{A}\cdot(\mathbf{1}+\mathbf{K}^{[N_{\text{val}}]})\|_{\mathcal{Y}^1}$ is the approximation error because A was (maybe) not the exact representation matrix of $(\mathbf{1}+\mathbf{K}^{[N_{\text{val}}]})^{-1}$.
- $\|\mathbf{A}\cdot(\mathbf{K}-\mathbf{K}^{[N_{\text{val}}]})\|_{\mathcal{Y}^1}$ is the truncation error because $\mathbf{K}^{[N_{\text{val}}]}$ is not exactly $\mathbf{K}$.

Section 4.3.1 focuses on the truncation error, which is tightly bounded by some rather technical inequalities, summarized in Algorithm IntopTruncError. The more straightforward computation of the approximation error is directly included in Algorithm IntopContract.

Once we have obtained a quasi-Newton operator $\mathbf{T}$ with a certified Lipschitz constant $\mu < 1$, the validation of a candidate solution φ° is summarized in Section 4.3.2, together with its complexity analysis.

4.3.1 ▶ Bounding the truncation error

The truncation error is computed by providing an upper bound for $\sup_{i \geqslant 0} B(i)$ where $B(i) := \|\mathbf{A} \cdot (\mathbf{K} - \mathbf{K}^{[N_{\mathrm{val}}]}) \cdot T_i\|_{\mathrm{U}^1}$. The indices i are divided into four groups:
– For $i \in [\![0, N_{\mathrm{val}} - d]\!]$, $\mathbf{K}^{[N_{\mathrm{val}}]} \cdot T_i = \mathbf{K} \cdot T_i$ (Lemma 4.12) and hence $B(i) = 0$.
– For $i \in [\![N_{\mathrm{val}} - d + 1, N_{\mathrm{val}}]\!]$, $\mathbf{A} \cdot (\mathbf{K} - \mathbf{K}^{[N_{\mathrm{val}}]}) \cdot T_i = (\mathbf{1} - \boldsymbol{\Pi}_{N_{\mathrm{val}}}) \cdot \mathbf{K} \cdot T_i$ are explicitly computed.
– For $i \in [\![N_{\mathrm{val}} + 1, N_{\mathrm{val}} + d]\!]$, $B(i) = \|\mathbf{A} \cdot \mathbf{K} \cdot T_i\|_{\mathrm{U}^1}$ and some of the diagonal coefficients of $\mathbf{K} \cdot T_i$ are of index less than N_{val} and are therefore non-trivially affected by $\mathbf{A}$. We choose to explicitly compute all these $\mathbf{A} \cdot \mathbf{K} \cdot T_i$.
– For $i > N_{\mathrm{val}} + d$, $(\mathbf{K} - \mathbf{K}^{[N_{\mathrm{val}}]}) \cdot T_i = \mathbf{K} \cdot T_i$ and the diagonal coefficients of $\mathbf{K} \cdot T_i$ are all located at indices strictly greater than N_{val}. We have $B(i) = B_I(i) + B_D(i)$ with:

- $B_D(i) := \|(\mathbf{1} - \boldsymbol{\Pi}_{N_{\mathrm{val}}}) \cdot \mathbf{K} \cdot T_i\|_{\mathrm{U}^1}$ due to diagonal coefficients, which decrease in $\mathcal{O}(1/i)$ from Equation (4.6).
- $B_I(i) := \|A \cdot \boldsymbol{\Pi}_{N_{\mathrm{val}}} \cdot \mathbf{K} \cdot T_i\|_{\mathrm{U}^1}$ due to initial coefficients multiplied by A, which decrease in $\mathcal{O}(1/i^2)$ from Equation (4.7).

The main difficulty is to bound $B(i)$ for $i > N_{\mathrm{val}} + d$, since we deal with an infinite number of indices i. For that, a natural idea is to use the explicit expression (4.8), replace i by the interval $[N_{\mathrm{val}} + d + 1, +\infty)$ and evaluate $\mathbf{A} \cdot \mathbf{K} \cdot T_i$ in interval arithmetics. Since these evaluations often lead to overestimations, one needs to choose a large value for N_{val}, such that the convergence in $\mathcal{O}(1/N_{\mathrm{val}})$ is sufficiently small to compensate. Usually, the chosen N_{val} is far larger than the one needed for $\mathbf{T}$ to be contracting.

A better solution consists in computing $\mathbf{A} \cdot \mathbf{K} \cdot T_{i_0}$ where $i_0 > N_{\mathrm{val}} + d$ and bounding the difference between $B(i)$ and $B(i_0)$ for all the remaining indices $i \geqslant i_0$.

Lemma 4.18 *Let $i \geqslant i_0 > N_{\mathrm{val}} + d$. Then*

(4.12 i) *For the diagonal coefficients, we have*

$$B_D(i) \leqslant B_D(i_0) + \frac{r \sum_{j=0}^{r-1} \|b_j\|_{\mathrm{U}^1}}{(i_0 - r)^2}.$$

(4.12 ii) *For the initial coefficients, we have*

$$B_I(i) \leqslant B_I(i_0) + \frac{r^3 \sum_{j=0}^{r-1} \|\mathbf{A} \cdot b_j\|_{\mathrm{U}^1}}{(i_0^2 - r^2)^2}.$$

Proof. For (4.12 i), from (4.8) we know that the diagonal coefficients of $\mathbf{K}\cdot T_i$, and respectively $\mathbf{K}\cdot T_{i_0}$, are those of the polynomials $\sum_{0\leqslant j<r}\sum_{-d_j\leqslant k\leqslant d_j} b_{jk}\gamma_{ij(i+k)}$ and $\sum_{0\leqslant j<r}\sum_{-d_j\leqslant k\leqslant d_j} b_{jk}\gamma_{i_0j(i_0+k)}$, respectively. All these coefficients are of positive index, so that we can shift them by $i-i_0$ positions to the right by replacing $\gamma_{i_0j(i_0+k)}$ with $\gamma_{i_0j(i+k)}$ without changing the norm (modifying the third index of $\gamma_{i_0j(i_0+k)}$ has no influence on the four coefficients of (4.6)). This ruse allows us to compare polynomials of equal degree i.e., $\gamma_{ij(i+k)}$ and $\gamma_{i_0j(i+k)}$:

$$\begin{aligned}|iB_D(i)-i_0B_D(i_0)| &= \left| i\left\|\sum_{j=0}^{r-1}\sum_{k=-d_j}^{d_j} b_{jk}\gamma_{ij(i+k)}\right\|_{\mathrm{Y}^1} - i_0\left\|\sum_{j=0}^{r-1}\sum_{k=-d_j}^{d_j} b_{jk}\gamma_{i_0j(i+k)}\right\|_{\mathrm{Y}^1}\right| \\ &\leqslant \sum_{j=0}^{r-1}\sum_{k=-d_j}^{d_j} |b_{jk}|\|i\gamma_{ij(i+k)}-i_0\gamma_{i_0j(i+k)}\|_{\mathrm{Y}^1}.\end{aligned}$$

Using the fact that for all x such that $|x|<i_0\leqslant i$,

$$\left|\frac{i}{i+x}-\frac{i_0}{i_0+x}\right| \leqslant \frac{i_0}{(i_0-|x|)^2}|x|,$$

we get that for any ℓ, $\|i\gamma_{ij\ell}-i_0\gamma_{i_0j\ell}\|_{\mathrm{Y}^1}\leqslant ri_0/(i_0-r)^2$. We conclude by noticing that

$$B_D(i)\leqslant \frac{i}{i_0}B_D(i)\leqslant B_D(i_0)+\frac{1}{i_0}|iB_D(i)-i_0B_D(i_0)|\leqslant B_D(i_0)+\frac{r\sum_{j=0}^{r-1}\|b_j\|_{\mathrm{Y}^1}}{(i_0-r)^2}.$$

For (4.12 ii), we have that

$$\begin{aligned}|i^2B_I(i)-i_0^2B_I(i_0)| &= \left| i^2\left\|\mathbf{A}\cdot\sum_{j=0}^{r-1}\gamma_{iji}(-1)b_j\right\|_{\mathrm{Y}^1} - i_0^2\left\|\mathbf{A}\cdot\sum_{j=0}^{r-1}\gamma_{i_0ji}(-1)b_j\right\|_{\mathrm{Y}^1}\right| \\ &\leqslant \sum_{j=0}^{r-1}\|\mathbf{A}\cdot b_j\|_{\mathrm{Y}^1}|i^2\gamma_{iji}(-1)-i_0^2\gamma_{i_0ji}(-1)|.\end{aligned}$$

We conclude using (4.7) and a similar inequality:

$$\left|\frac{i^2}{i^2-x^2}-\frac{i_0^2}{i_0^2-x^2}\right|\leqslant\frac{i_0^2}{(i_0^2-x^2)^2}x^2.$$

□

In practice, this method yields more accurate bounds when the parameters of the problem become somehow large. This is due to the fact that the part potentially affected by overestimations is divided by greater power of i_0 (i_0^2 and i_0^4) than in the previously mentioned method (i_0 and i_0^2).

Note that the bounds announced by **Lemma 4.18** can be sharpened if we don't replace $|j\pm 1|$ with r. The obtained formulas are essentially not more difficult to implement, but we omit these details for the sake of clarity.

Algorithm 4.6 IntopTruncError$(\mathbf{K}, N_{\text{val}}, A)$ – Bounding the truncation error

Input: a polynomial integral operator $\mathbf{K}$ (given by its order r and the $b_j(t)$), a truncation order N_{val} and an approximate inverse A of $\mathbf{1} + \mathbf{K}^{[N_{\text{val}}]}$.

Output: an upper bound δ_{trunc} for the truncation error $\|\mathbf{A} \cdot (\mathbf{K} - \mathbf{K}^{[N_{\text{val}}]})\|_{\mathfrak{U}^1}$.

▷ ***All operations are to be performed in interval arithmetics***

▷ *Compute* $\delta_{trunc}^{(1)} \geqslant \sup_{i \in [\![N_{\text{val}}-d+1, N_{\text{val}}]\!]} B(i)$

$\delta_{trunc}^{(1)} \leftarrow 0$
for $i = N_{\text{val}} - d + 1$ **to** N_{val} **do**
  $P \leftarrow (\mathbf{1} - \boldsymbol{\pi}_{N_{\text{val}}}) \cdot \mathbf{K} \cdot T_i$
  if $\|P\|_{\mathfrak{U}^1} > \delta_{trunc}^{(1)}$ **then** $\delta_{trunc}^{(1)} \leftarrow \|P\|_{\mathfrak{U}^1}$
end for

▷ *Compute* $\delta_{trunc}^{(2)} \geqslant \sup_{i \in [\![N_{\text{val}}+1, N_{\text{val}}+d]\!]} B(i)$

$\delta_{trunc}^{(2)} \leftarrow 0$
for $i = N_{\text{val}} + 1$ **to** $N_{\text{val}} + d$ **do**
  $P \leftarrow \mathbf{A} \cdot \mathbf{K} \cdot T_i$
  if $\|P\|_{\mathfrak{U}^1} > \delta_{trunc}^{(2)}$ **then** $\delta_{trunc}^{(2)} \leftarrow \|P\|_{\mathfrak{U}^1}$
end for

▷ *Compute* $\delta_{trunc}^{(3)} \geqslant \sup_{i \geqslant N_{\text{val}}+d+1} B_D(i)$

$i_0 \leftarrow N_{\text{val}} + d + 1$ and $B \leftarrow \sum_{j=0}^{r-1} \|b_j\|_{\mathfrak{U}^1}$
$P \leftarrow (\mathbf{1} - \boldsymbol{\pi}_{N_{\text{val}}}) \cdot \mathbf{K} \cdot T_{i_0}$ and $\delta_{trunc}^{(3)} \leftarrow \|P\|_{\mathfrak{U}^1}$
$\delta_{trunc}^{(3)} \leftarrow \delta_{trunc}^{(3)} + \frac{rB}{(i_0 - r)^2}$

▷ *Compute* $\delta_{trunc}^{(4)} \geqslant \sup_{i \geqslant N_{\text{val}}+d+1} B_I(i)$

$B \leftarrow \sum_{j=0}^{r-1} \|\mathbf{A} \cdot b_j\|_{\mathfrak{U}^1}$
$P \leftarrow \mathbf{A} \cdot \boldsymbol{\pi}_{N_{\text{val}}} \cdot \mathbf{K} \cdot T_{i_0}$ and $\delta_{trunc}^{(4)} \leftarrow \|P\|_{\mathfrak{U}^1}$
$\delta_{trunc}^{(4)} \leftarrow \delta_{trunc}^{(4)} + \frac{r^3 B}{(i_0^2 - r^2)^2}$

$\delta_{trunc} \leftarrow \max(\delta_{trunc}^{(1)}, \delta_{trunc}^{(2)}, \delta_{trunc}^{(3)} + \delta_{trunc}^{(4)})$
return δ_{trunc}

■ **Proposition 4.19** *Algorithm* IntopTruncError *is correct and requires* $\mathcal{O}((h'+d')(h+d)d)$ *operations when* A *is* (h',d') *almost-banded, or* $\mathcal{O}(N_{\text{val}}(h+d)d)$ *operations when* A *is dense.*

Proof. The correctness is straightforward, using **Lemma 4.18**. To reach the claimed complexity, the polynomials $\mathbf{K}\cdot T_i$ involved in the algorithm must be sparsely computed as an (h,d) almost-banded vector around index i, using $\mathcal{O}(rh)$ arithmetic operations. Clearly, step 1 for $\delta^{(1)}_{trunc}$ (lines 1-5) costs $\mathcal{O}(drh)$ operations. For each i in step 2 for $\delta^{(2)}_{trunc}$ (lines 6-10), computing $\mathbf{K}\cdot T_i$ costs $\mathcal{O}(rh)$ operations to obtain an (h,d) almost-banded vector, and applying $\mathbf{A}$ costs $\mathcal{O}((h'+d')(h+d))$ or $\mathcal{O}(N_{\text{val}}(h+d))$ operations, depending on whether A is (h',d') almost-banded or dense (see **Table 4.1** in **Section 4.2**). Hence we get $\mathcal{O}((h'+d')(h+d)d)$ or $\mathcal{O}(N_{\text{val}}(h+d)d)$ operations. After that, step 3 for $\delta^{(3)}_{trunc}$ (lines 11-13) costs $\mathcal{O}(rh)$ operations both to compute $(\mathbf{1}-\boldsymbol{\pi}_{N_{\text{val}}})\cdot\mathbf{K}\cdot T_{i_0}$ and $\sum_{j=0}^{r-1}\|b_j\|_{\mathrm{U}^1}$. Finally, at step 4 (lines 14-16), computing $\sum_{j=0}^{r-1}\|A\cdot b_j\|_{\mathrm{U}^1}$ costs $\mathcal{O}((h'+d')rh)$ or $\mathcal{O}(N_{\text{val}}rh)$ operations, and computing $A\cdot\boldsymbol{\pi}_{N_{\text{val}}}\cdot\mathbf{K}\cdot T_{i_0}$ costs $\mathcal{O}(rh+(h'+d')(h+d))$ or $\mathcal{O}(rh+N_{\text{val}}(h+d))$ operations. We see that in both cases (A (h',d') almost-banded or dense), the most expensive step is the second one, which gives the respective expected total complexities. □

4.3.2 ▶ Complete validation method and complexity

We now have all the ingredients for the complete validation process: **Algorithm** IntopContract obtains a contracting Newton-like operator $\mathbf{T}$ and **Algorithm** IntopVal validates a candidate solution φ°.

For **Algorithm** IntopContract, the parameters h, d and the $\|b_j\|_{\mathrm{U}^1}$ directly come from LODE (4.2), while the other input parameters N_{val}, h' and d' must either be known by the user or obtained from a decision procedure. For that, first, **Proposition 4.20** analyses the complexity of IntopContract and **Algorithm** IntopVal when N_{val}, h' and d' are given. Then, a more detailed study of the magnitude of these parameters and an intuition on how to choose them is proposed.

In ChebValid, the functions and auxiliary routines implementing **Algorithms** IntopContract and IntopVal are located in `mpfi_chebpoly_intop_newton.h`.

■ Complexity in function of the chosen parameters

■ **Proposition 4.20** *Let* $\mathbf{K}$ *be the integral operator associated to the polynomial LODE (4.2),* N_{val} *be the truncation order chosen for the quasi-Newton method,* $M=\mathbf{1}+\mathbf{K}^{[N_{\text{val}}]}$ *and* (h,d) *the parameters of its almost-banded structure,* A *the approximation of* M^{-1} *used for* $\mathbf{T}$*, either dense or* (h',d') *almost-banded. We have the following complexity results:*

(4.20 i) *The complexity of producing the Newton-like operator* $\mathbf{T}$ *and validating its* U^1*-norm using* ***Algorithm*** IntopContract *is:*

$$\mathcal{O}(N_{\text{val}}(h+d)(h'+d')) \quad (\text{or } \mathcal{O}({N_{\text{val}}}^2(h+d)) \text{ when } A \text{ is dense}).$$

Algorithm 4.7 IntopContract($\mathbf{K}, N_{\text{val}}, h', d'$) – Creating and bounding a Newton-like op. **T**

Input: a polynomial integral operator $\mathbf{K}$ (given by its order r and the $b_j(t)$), a truncation order N_{val} and optional parameters h' and d'.

Output: an approximate inverse A of $\mathbf{1} + \mathbf{K}^{[N_{\text{val}}]}$ ((h', d') almost-banded if h' and d' were specified, dense otherwise) and a certified Lipschitz constant μ.

$M \leftarrow \mathbf{1} + \mathbf{K}^{[N_{\text{val}}]}$, computed as an (h, d) almost-banded matrix.
$(Q, R) \leftarrow$ OlverTownsendQR(M)

▷ *Compute the approximate inverse A of M*
if h' and d' are specified with $h \leqslant h' < N_{\text{val}}$ and $d \leqslant d' < N_{\text{val}}$ **then**
  ▷▷ *A is (h', d') almost-banded*
  $A \leftarrow$ AlmostBandedApproxInverse($M, (Q, R), h', d'$)
else
  ▷▷ *A is dense*
  for $i = 0$ **to** $N_{\text{val}} - 1$ **do**
    for $j = 0$ **to** $N_{\text{val}} - 1$ **do** $V[j] \leftarrow 0$ and $V[i] \leftarrow 1$
    $W \leftarrow$ OlverTownsendBackSubs($M, (Q, R), V$)
    Set ith column of A to W
  end for
end if

▷ *Compute the approximation error $\delta_{approx} \geqslant \|\mathbf{1} - \mathbf{A} \cdot (\mathbf{1} + \mathbf{K}^{[N_{\text{val}}]})\|_{\mathcal{U}^1}$*
$\delta_{approx} \leftarrow 0$
for $i = 0$ **to** $N_{\text{val}} - 1$ **do**
  Set V to the ith column of M, as an (h, d) almost-banded vector
  Compute $W \leftarrow A \cdot V$ and $W[i] \leftarrow W[i] - 1$ with interval arithmetics
  if $\|W\|_1 > \delta_{approx}$ **then** $\delta_{approx} \leftarrow \|W\|_1$
end for
▷ *Compute the truncation error $\delta_{trunc} \geqslant \|\mathbf{A} \cdot (\mathbf{K} - \mathbf{K}^{[N_{\text{val}}]})\|_{\mathcal{U}^1}$*
$\delta_{trunc} \leftarrow$ IntopTruncError($\mathbf{K}, N_{\text{val}}, A$)

$\mu \leftarrow \delta_{approx} + \delta_{trunc}$
if $\mu < 1$ **then**
  return μ
else
  return "Fail"
end if

Algorithm 4.8 IntopVal$(\mathbf{K}, \psi, N_{\mathrm{val}}, A, \mu, \varphi^\circ)$ – Validating a solution of integral equation

Input: a polynomial integral operator $\mathbf{K}$ (given by its order r and the $b_j(t)$), a polynomial right-hand side ψ, a truncation order N_{val}, (A, μ), with $\mu < 1$, computed by IntopContract$(\mathbf{K}, N_{\mathrm{val}}, h', d')$, and a candidate solution φ° of degree N_{app}.

Output: an error bound ε such that $\|\varphi^\circ - \varphi^*\|_{\mathrm{Y}^1} \leqslant \varepsilon$.

▷ ***All operations are to be performed in interval arithmetics***

1: $P \leftarrow \varphi^\circ + \mathbf{K} \cdot \varphi^\circ - \psi$
2: **for** $i = 0$ **to** N_{val} **do** $V[i] \leftarrow [P]_i$
3: $W \leftarrow AV$
4: **for** $i = 0$ **to** N_{val} **do** $[P]_i \leftarrow W[i]$
5: $\varepsilon \leftarrow \|P\|_{\mathrm{Y}^1}/(1-\mu)$
6: **return** ε

(4.20 ii) *Having this validated Newton-like operator, a degree N_{app} approximate solution φ° of (4.2) (with $N_{\mathrm{rhs}} = \deg \psi$) is validated using **Algorithm** IntopVal in:*

$$\mathcal{O}(N_{\mathrm{app}} r h + N_{\mathrm{rhs}} + (h' + d') \min(N_{\mathrm{val}}, \max(N_{\mathrm{app}} + d, N_{\mathrm{rhs}})))$$
$$(\textit{or } \mathcal{O}(N_{\mathrm{app}} r h + N_{\mathrm{rhs}} + N_{\mathrm{val}} \min(N_{\mathrm{val}}, \max(N_{\mathrm{app}} + d, N_{\mathrm{rhs}}))) \textit{ when } A \textit{ is dense}).$$

Proof. For (4.20 i), we consider the different steps to obtain $\mathbf{T}$ and bound its Y^1-norm:

- Computing $M = \mathbf{1} + \mathbf{K}^{[N_{\mathrm{val}}]}$ (line 1) costs $\mathcal{O}(N_{\mathrm{val}} r h)$ operations, using the defining formula (4.5) of $\mathbf{K}$, and $\mathcal{O}(N_{\mathrm{val}} d^2)$ operations are needed for the QR decomposition (line 2) according to **Proposition 4.14**.
- Computing A (lines 3-11) needs $\mathcal{O}(N_{\mathrm{val}}(h+d)(h'+d'))$ operations in the almost-banded case (**Corollary 4.17**), or $\mathcal{O}({N_{\mathrm{val}}}^2(h+d))$ in the dense case.
- Using **Table 4.1**, line 15 costs $\mathcal{O}((h'+d')(h+d))$ operations when A is (h', d') almost-banded, or $\mathcal{O}(N_{\mathrm{val}}(h+d))$ when it is dense. Hence the computation of the approximation error is performed in $\mathcal{O}(N_{\mathrm{val}}(h'+d')(h+d))$ (almost-banded case) or $\mathcal{O}({N_{\mathrm{val}}}^2(h+d))$ (dense case) operations.
- The truncation error (line 18) costs $\mathcal{O}((h'+d')(h+d)d)$ operations when A is (h', d') almost-banded, or $\mathcal{O}(N_{\mathrm{val}}(h+d)d)$ in the dense case, following **Proposition 4.19**.

Hence, the total complexity is in $\mathcal{O}(N_{\mathrm{val}}(h+d)(h'+d'))$ when A is (h', d') almost-banded, or $\mathcal{O}({N_{\mathrm{val}}}^2(h+d))$ when A is dense.

For (4.20 ii), computing P (of degree $\max(N_{\mathrm{app}}+d, N_{\mathrm{rhs}})$) at line 1 costs $\mathcal{O}(N_{\mathrm{app}} r h + N_{\mathrm{rhs}})$ operations. Multiplying by A its $n+1$ first coefficients (line 3) requires $\mathcal{O}((h'+d')\min(N_{\mathrm{val}}, \max(N_{\mathrm{app}}+d, N_{\mathrm{rhs}}))$ operations (if A (h', d') almost-banded) or $\mathcal{O}(N_{\mathrm{val}} \min(N_{\mathrm{val}}, \max(N_{\mathrm{app}} + d, N_{\mathrm{rhs}})))$ operations (if A dense). Note that at line 2, copying the $N_{\mathrm{val}}+1$ first coefficients of P costs $\min(\max(N_{\mathrm{app}}+d, N_{\mathrm{rhs}}), N_{\mathrm{val}})$ (neglect the null coefficients), and in the almost-banded case when $\max(N_{\mathrm{app}}+d, N_{\mathrm{rhs}}) < N_{\mathrm{val}}$, lines 4 costs $(\max(N_{\mathrm{app}}+d, N_{\mathrm{rhs}}) + h' + d')$ operations (again, neglect the final null coefficients). This yields the claimed total complexity. □

Choosing and estimating parameters N_{val}, h' and d'

The complexity claimed by **Proposition 4.20** depends on the parameters N_{val}, h' and d'. Hence, the performance of the validation method is directly linked to the minimal values we can choose for these parameters.

In practice, one initializes $N_{\mathrm{val}} = 2d$ (to avoid troubles with too small values of N_{val}) and then estimates (from below) the norm $\|(\mathbf{1} + \mathbf{K}^{[N_{\mathrm{val}}]})^{-1} \cdot (\mathbf{K} - \mathbf{K}^{[N_{\mathrm{val}}]})\|_{\mathrm{U}^1}$ by numerically applying this operator on $T_{N_{\mathrm{val}}+1}$. This heuristic is similar to estimating the truncation error of a Chebyshev series by its first neglected term **[53, §4.4, Thm. 6]**[2]. Specifically, for intermediate or large values of N_{val}, one has for $i \leqslant N_{\mathrm{val}}$ that $\|(\mathbf{K} - \mathbf{K}^{[N_{\mathrm{val}}]}) \cdot T_i\|_{\mathrm{U}^1} \leqslant \|\mathbf{K} \cdot T_i\|_{\mathrm{U}^1}$, and for $i \geqslant N_{\mathrm{val}}+1$, one has a decrease of $\|\mathbf{K} \cdot T_i\|_{\mathrm{U}^1}$ in $\mathcal{O}(1/i)$. Recall that for $i \geqslant N_{\mathrm{val}}$, $\mathbf{K}^{[N_{\mathrm{val}}]} \cdot T_i = 0$, from **(4.9)**. So, $\max_{i \geqslant 0} \|(\mathbf{1}+\mathbf{K}^{[N_{\mathrm{val}}]})^{-1} \cdot (\mathbf{K}-\mathbf{K}^{[N_{\mathrm{val}}]}) \cdot T_i\|_{\mathrm{U}^1}$ is heuristically achieved for $i = N_{\mathrm{val}}+1$. Concretely, computing $\|(\mathbf{1}+\mathbf{K}^{[N_{\mathrm{val}}]})^{-1} \cdot (\mathbf{K}-\mathbf{K}^{[N_{\mathrm{val}}]}) \cdot T_{N_{\mathrm{val}}+1}\|_{\mathrm{U}^1}$ reduces to numerically solving the corresponding almost-banded system with input parameters M and $\boldsymbol{\pi}_{N_{\mathrm{val}}} \cdot K \cdot T_{N_{\mathrm{val}}+1}$ using **Algorithms** OlverTownsendQR and OlverTownsendBackSubs.

If this estimate from below of the norm of $\mathbf{T}$ is greater than 1, we double the value of N_{val} until the estimated norm falls below 1. Then we initialize $h' = h$ and $d' = d$, compute an (h', d') almost-banded approximation of $(\mathbf{1} + \mathbf{K}^{[N_{\mathrm{val}}]})^{-1}$ using **Algorithm** AlmostBandedApproxInverse and double their values each time the approximation error exceeds 0.25. After that, **Algorithm** IntopTruncError produces a certified upper bound for the truncation error. If it exceeds 0.25, then again we double the value of N_{val} and restart the validation process.

In what follows, we give theoretical estimates for the order of magnitude of the above mentioned parameters. First a bound for N_{val} is

$$N_{\mathrm{val}} = \mathcal{O}(dB^2 \exp(2B)), \quad \text{where } B = \sum_{j=0}^{r-1} \|b_j\|_{\mathrm{U}^1}.$$

This can be proved since N_{val} must be chosen large enough so that the sum of the approximation and truncation errors falls below 1. For this, a sufficient condition is $\|(\mathbf{1} + \mathbf{K})^{-1} \cdot (\mathbf{K} - \mathbf{K}^{[N_{\mathrm{val}}]})\|_{\mathrm{U}^1} < 1$, using the proof of **Lemma (4.13 i)**, **(4.13 ii)** and **[93, Chap. 2, Cor. 8.2]**. The estimate follows since $\|\mathbf{K} - \mathbf{K}^{[N_{\mathrm{val}}]}\|_{\mathrm{U}^1} = \mathcal{O}(B/N_{\mathrm{val}})$, from **Lemma 4.12** and

$$\|(\mathbf{1} + \mathbf{K})^{-1}\|_{\mathrm{U}^1} \leqslant \sum_{i=0}^{+\infty} (6di + 1) \frac{2C}{i!} \leqslant (12dB + 1) \exp(2B),$$

using **Lemma 4.10** and the fact that C (defined in **(4.4)**) is upper bounded by B.

Now, for the almost-banded parameters h', d', we provide a practical estimate of

$$h', d' = \mathcal{O}(dB).$$

This is based on the observation that for sufficiently large N_{val}, we can expect the ℓth iterated operator $(\mathbf{K}^{[N_{\mathrm{val}}]})^\ell$ to behave approximately like $\mathbf{K}^\ell$. Since $\|\mathbf{K}^\ell\|_{\mathrm{U}^1} \leqslant (6d\ell + 1)(2B)^\ell/\ell!$ (proof of **Lemma 4.10**), this quantity falls below 1 as soon as $\ell \approx 2B \exp(1)$. Then $(\mathbf{1}+\mathbf{K}^{[N_{\mathrm{val}}]})^{-1} = \sum_{i=0}^{+\infty} (-\mathbf{K}^{[N_{\mathrm{val}}]})^i$, and $\mathbf{A} = \sum_{i=0}^{\ell-1} (-\mathbf{K}^{[N_{\mathrm{val}}]})^i$ is $(d(\ell-1), d(\ell-1))$ almost-banded (again in proof

[2] **[34, §2.12]** presents, as a rule-of-thumb, the estimate of the truncation error by the last term retained.

of **Lemma 4.10**). We therefore obtain an approximation error $\|\mathbf{1} - \mathbf{A}(\mathbf{1} + \mathbf{K}^{[N_{\text{val}}]})\|_{\text{Ч}^1} = \|(\mathbf{K}^{[N_{\text{val}}]})^\ell\|_{\text{Ч}^1} < 1$.

To conclude, although it provides a rigorous complexity estimate, the bound concerning N_{val} is usually very pessimistic. This is because the above mentioned practical approach of doubling N_{val} ends up with far smaller values in most cases. It often happens that $\|(\mathbf{1}+\mathbf{K})^{-1}\|_{\text{Ч}^1}$ does not follow an exponential growth when $B = \sum_{j=0}^{r-1} \|b_j\|_{\text{Ч}^1}$ becomes large. For instance, when $k(t,s)$ is nonnegative, then the Neumann series $\sum_{i=0}^{+\infty} \mathbf{K}^i$ (equal to $\mathbf{1}+\mathbf{K}$) alternates signs and the Ч^1-norm of $(\mathbf{1}+\mathbf{K})^{-1}$ is far smaller than the term-by-term exponential bound. Several examples in **Section 4.5** illustrate this phenomenon. In the difficult cases involving an exponential growth of $\|(\mathbf{1}+\mathbf{K})^{-1}\|_{\text{Ч}^1}$, the examples in **Section 4.5** also show how the almost-banded approach helps to keep the computation tractable up to some extent.

4.4 Extensions to non-polynomial LODEs

In this section, we show how to address the general case stated in **Problem 4.1**. In **Section 4.4.1** we extend the previously described method to the non-polynomial case with Cauchy boundary conditions. Then we discuss the case of other boundary conditions in **Section 4.4.2**.

4.4.1 ▶ Extension to non-polynomial IVP

We consider the IVP problem (1a) where the coefficients α_j, $j \in [\![0, r-1]\!]$, and the right-hand side γ belong to Ч^1 and are rigorously approximated by Chebyshev models $\mathfrak{a}_j = (a_j, \varepsilon_j)$ and $\mathfrak{b} = (g, \tau)$. Using **Proposition 4.5**, we get an integral operator $\mathbf{K}$ with a kernel $k(t,s)$ which is polynomial in the variable s only:

$$k(t,s) = \sum_{j=0}^{r-1} \beta_j(t) T_j(s),$$

where the β_j are non-polynomial functions in Ч^1.

To obtain Chebyshev models $\mathfrak{B}_j = (b_j, \eta_j)$ for β_j it suffices to run **Algorithm** IntegralTransform where one replaces the polynomials a_j by Chebyshev models $\mathfrak{a}_j$ and overloads corresponding arithmetic operations. Then, the polynomials b_j define a polynomial kernel $k_P(t,s)$ as in Equation (4.5) and respectively the polynomial integral operator $\mathbf{K}_P$, such that:

$$\|\mathbf{K} - \mathbf{K}_P\|_{\text{Ч}^1} \leqslant 2 \sum_{j=0}^{r-1} \eta_j. \tag{4.14}$$

Moreover, since **Algorithm** IntegralTransform only performs linear operations on the Chebyshev models $\mathfrak{a}_j$ to produce the $\mathfrak{B}_j$, the quantity $\sum_{0\leqslant j<r} \eta_j$ is upper bounded by $C \sum_{0\leqslant j<r} \varepsilon_j$ for some constant C depending only on r. This justifies the fact that $\mathbf{K}$ is well approximated by $\mathbf{K}_P$ when the coefficients α_j are well approximated by the a_j.

Let us prove that the truncated operators $\mathbf{K}^{[n]} := \boldsymbol{\pi}_n \cdot \mathbf{K} \cdot \boldsymbol{\pi}_n$ still converge to $\mathbf{K}$ and that $\mathbf{1} + \mathbf{K}$ is an isomorphism of $\mathcal{Y}^1$:

Lemma 4.21 *Let $\mathbf{K}$ be the integral operator obtained from **Proposition 4.5**. We have*

(4.21 i) *$\mathbf{K}$ is a bounded linear operator of $\mathcal{Y}^1$ with:*

$$\|\mathbf{K}\|_{\mathcal{Y}^1} \leqslant 2 \sum_{j=0}^{r-1} \|\beta_j\|_{\mathcal{Y}^1}.$$

(4.21 ii) *$\mathbf{K}^{[n]} \to \mathbf{K}$ for the $\mathcal{Y}^1$-operator norm as $n \to +\infty$. Hence $\mathbf{K}$ is compact.*

(4.21 iii) *$\mathbf{1} + \mathbf{K}$ is a bicontinuous isomorphism of $\mathcal{Y}^1$.*

Proof. For (4.21 i), let $\varphi \in \mathcal{Y}^1$. From (2.10) and (2.12), we have

$$\|\mathbf{K} \cdot \varphi\|_{\mathcal{Y}^1} \leqslant \sum_{j=0}^{r-1} \|\beta_j\|_{\mathcal{Y}^1} (2\|T_j\|_{\mathcal{Y}^1} \|\varphi\|_{\mathcal{Y}^1}) = \left(2 \sum_{j=0}^{r-1} \|\beta_j\|_{\mathcal{Y}^1} \right) \|\varphi\|_{\mathcal{Y}^1}.$$

This shows that $\mathbf{K} \cdot \varphi \in \mathcal{Y}^1$ and that $\mathbf{K}$ is bounded as endomorphism of $(\mathcal{Y}^1, \|\cdot\|_{\mathcal{Y}^1})$ with the bound claimed above.

For (4.21 ii), let $\varepsilon > 0$. Take Chebyshev models $\mathfrak{a}_j = (a_j, \varepsilon_j)$ of α_j sufficiently accurate to ensure $\|\mathbf{K} - \mathbf{K}_P\|_{\mathcal{Y}^1} \leqslant \varepsilon/3$, by (4.14). This is possible since the α_j belong to $\mathcal{Y}^1$, and hence the η_j can be made as small as desired. We know from **Lemma 4.12**, since $\mathbf{K}_P$ is polynomial, that for n large enough, $\|\mathbf{K}_P - \mathbf{K}_P^{[n]}\|_{\mathcal{Y}^1} \leqslant \varepsilon/3$. We finally get:

$$\begin{aligned} \|\mathbf{K} - \mathbf{K}^{[n]}\|_{\mathcal{Y}^1} &\leqslant \|\mathbf{K} - \mathbf{K}_P\|_{\mathcal{Y}^1} + \|\mathbf{K}_P - \mathbf{K}_P^{[n]}\|_{\mathcal{Y}^1} + \|\mathbf{K}_P^{[n]} - \mathbf{K}^{[n]}\|_{\mathcal{Y}^1} \\ &\leqslant \|\mathbf{K} - \mathbf{K}_P\|_{\mathcal{Y}^1} + \|\mathbf{K}_P - \mathbf{K}_P^{[n]}\|_{\mathcal{Y}^1} + \|\mathbf{K}_P - \mathbf{K}\|_{\mathcal{Y}^1} \\ &\leqslant \frac{\varepsilon}{3} + \frac{\varepsilon}{3} + \frac{\varepsilon}{3} = \varepsilon, \end{aligned}$$

where we used that $\|\mathbf{K}_P^{[n]} - \mathbf{K}^{[n]}\|_{\mathcal{Y}^1} = \|\boldsymbol{\pi}_n \cdot (\mathbf{K}_P - \mathbf{K}) \cdot \boldsymbol{\pi}_n\|_{\mathcal{Y}^1} \leqslant \|\mathbf{K}_P - \mathbf{K}\|_{\mathcal{Y}^1}$.

For (4.21 iii), the proof works exactly as in the polynomial case: we know that $\mathbf{K}$ is compact by 2) and that $\mathbf{1}+\mathbf{K}$ is injective because it is injective over the superspace $\mathcal{C}^0$, cf. **Section 4.1.1**, and we conclude thanks to the Fredholm alternative. □

Using this result, we can again form the Newton-like operator $\mathbf{T}$ as in **Section 4.3**, with $\mathbf{A} \approx (\mathbf{1} + \mathbf{K}_P^{[n]})^{-1}$ for some large enough value of n.

The operator norm of the linear part of $\mathbf{T}$ can now be decomposed into three parts:

$$\|\mathbf{1} - \mathbf{A} \cdot (\mathbf{1} + \mathbf{K})\|_{\mathcal{Y}^1} \leqslant \|\mathbf{1} - \mathbf{A} \cdot (\mathbf{1} + \mathbf{K}_P^{[n]})\|_{\mathcal{Y}^1} + \|\mathbf{A} \cdot (\mathbf{K}_P - \mathbf{K}_P^{[n]})\|_{\mathcal{Y}^1} + \|\mathbf{A} \cdot (\mathbf{K} - \mathbf{K}_P)\|_{\mathcal{Y}^1}.$$

The first two parts are exactly the ones of (4.13) (where the polynomial integral operator $\mathbf{K}$ is now called $\mathbf{K}_P$) and can be rigorously upper bounded using the same techniques. The last part can be upper bounded thanks to (4.14):

$$\|\mathbf{A}\cdot(\mathbf{K}-\mathbf{K}_P)\|_{\mathcal{U}^1} \leqslant 2\|\mathbf{A}\|_{\mathcal{U}^1}\sum_{j=0}^{r-1}\eta_j. \tag{4.15}$$

It is interesting to notice that the order of magnitude of n is largely determined by the second part (as in the polynomial case), whereas the third part forces the η_j (and hence the ε_j) to be small, which mainly depends on the degree of the approximating polynomials $a_j(t)$ for $\alpha_j(t)$.

Finally, let φ° be the numerical approximation for the solution of the IVP problem (1a), given as a polynomial in the Chebyshev basis. One upper bounds $\|\mathbf{T}\cdot\varphi^\circ-\varphi^\circ\|_{\mathcal{U}^1} = \|\mathbf{A}\cdot(\varphi^\circ+\mathbf{K}\cdot\varphi^\circ-\psi)\|_{\mathcal{U}^1} \leqslant \|\mathbf{A}\cdot z\|_{\mathcal{U}^1}+\tau\|\mathbf{A}\|_{\mathcal{U}^1}$, where $\zeta=(z,\tau)$ is a Chebyshev model for $\varphi^\circ+\mathbf{K}\cdot\varphi^\circ-\psi$ obtained by arithmetic operations on RPAs.

Proposition 4.22 *The results of Proposition 4.20 remain valid for the IVP validation in the non-polynomial case (1a).*

Proof. For computing a rigorous Lipschitz constant for $\mathbf{T}$, the additional term $\|\mathbf{A}\cdot(\mathbf{K}-\mathbf{K}_P)\|_{\mathcal{U}^1}$ is bounded by (4.15). Clearly, this additional cost is dominated by the complexity obtained in Proposition 4.20 (i) for the polynomial case.

Then, validating a candidate solution φ° has the same cost as in the polynomial case (Proposition 4.20 (ii)), since all polynomial operations are essentially replaced by their Chebyshev model extensions. □

In conclusion, we observe that our validation method is easily adapted to the general case where the coefficients α_j are non-polynomial functions rigorously approximated by polynomials a_j. However, contrary to the polynomial case where the involved degrees are usually low, the degrees of the approximations a_j can be rather large, resulting in a dense linear problem and poorer time efficiency. And yet, in practice, the method remains efficient on problems with reasonable coefficient magnitude and time interval under consideration, which will be exemplified in Section 4.5.

The corresponding C routines are gathered in `chebmodel_intop_newton.h`. In particular, function `chebmodel_lode_intop_newton_solve_fr` completely solves Problem 4.1.

4.4.2 ▶ The case of other boundary conditions

Consider now the general boundary conditions operator $\mathbf{\Lambda} : \mathcal{U}^1 \to \mathbb{R}^r$ of Problem (1b). In [265] an *ad-hoc* integral reformulation is proposed to treat a specific case of such boundary conditions, while other works like [73] propose a generic reformulation method. Our method consists in reducing a general BVP validation problem to $r+1$ IVP validation problems. This is easily observed, since the initial values $v_j = f^{(j)}(t_0)$ appearing in the integral reformulation of Proposition 4.5 are now unknown. At first sight, this may seem rather naive and time-consuming. However, the most difficult part which consists in obtaining a contracting Newton-like operator is performed only once, thus considerably reducing the total computation time.

Suppose we have a candidate polynomial approximation f° of the solution of BVP problem (1b), given in Chebyshev basis. Our method consists in rigorously computing a very accurate approximation $\overline{f}$ and then comparing it with f°.

1. The first step is to provide $\mathbf{A}$ and compute an upper bound μ for $\|\mathbf{1} - \mathbf{A}\cdot(\mathbf{1}+\mathbf{K})\|_{\mathrm{q}^1}$. This depends neither on the initial conditions nor on the right hand side $\gamma(t)$.

2. Then, for each $i \in [\![0, r-1]\!]$, we compute (with **Algorithms** OlverTownsendQR and OlverTownsendBackSubs for the underlying linear algebra) and validate with **Algorithm** IntopVal an approximation f_i° for the solution f_i^* of the homogeneous LODE associated to (4.1) (that is, with right hand side $g = 0$) with initial conditions:

$$v_j = f_i^{(j)}(t_0) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases} \qquad 0 \leqslant j < r.$$

   Similarly, we approximate and validate the solution f_r^* of Equation (4.1) with right hand side g and null initial conditions ($f_r^{(j)} = 0$ for $0 \leqslant j < r$). Since the validation kernel has been produced at the previous step, the numerical solving procedure (**Algorithms** OlverTownsendQR and OlverTownsendBackSubs) as well as the validation (**Algorithm** IntopVal) are linear in the degree of the approximation. Thus, we obtain Chebyshev models for f_i^*, and for their derivatives $f_i^{*(j)}$, $0 \leqslant j \leqslant r$.

3. The original equation with boundary conditions $\mathbf{\Lambda}\cdot f = (\lambda_0(f), \dots, \lambda_{r-1}(f)) = (v_0, \dots, v_{r-1})$ admits a unique solution f^* if and only if there exist $c_0, c_1, \dots, c_{r-1}$ uniquely determined such that

$$f^* = c_0 f_0^* + c_1 f_1^* + \cdots + c_{r-1} f_{r-1}^* + f_r^*$$

   and

$$\begin{aligned} \lambda_0(f_0^*)c_0 + \lambda_0(f_1^*)c_1 + \cdots + \lambda_0(f_{r-1}^*)c_{r-1} &= -\lambda_0(f_r^*), \\ \lambda_1(f_0^*)c_0 + \lambda_1(f_1^*)c_1 + \cdots + \lambda_1(f_{r-1}^*)c_{r-1} &= -\lambda_1(f_r^*), \\ &\vdots \\ \lambda_{r-1}(f_0^*)c_0 + \lambda_{r-1}(f_1^*)c_1 + \cdots + \lambda_{r-1}(f_{r-1}^*)c_{r-1} &= -\lambda_{r-1}(f_r^*). \end{aligned}$$

   If the quantities $\lambda_j(f_i^*)$ can be rigorously and accurately computed using the Chebyshev models of the $f_i^{*(j)}$ obtained at the previous step, then one can solve this linear system in interval arithmetics [221] and obtain intervals $\mathbb{c}_0, \dots, \mathbb{c}_{r-1}$.

4. Using the (interval) coefficients $\mathbb{c}_0, \dots, \mathbb{c}_{r-1}$ and the Chebyshev models $\mathbb{f}_0, \dots, \mathbb{f}_{r-1}, \mathbb{f}_r$, we get that

$$\mathbb{f} := (\overline{f}, \varepsilon) := \mathbb{c}_0\mathbb{f}_0 + \cdots + \mathbb{c}_{r-1}\mathbb{f}_{r-1} + \mathbb{f}_r$$

   is a Chebyshev model for the exact solution f^*. Now, it suffices to compute $\eta = \|f^\circ - \overline{f}\|_{\mathrm{q}^1}$ (which is straightforward since both f° and $\overline{f}$ are polynomials in Chebyshev basis) and we deduce that the approximation error $\|f^\circ - f^*\|_{\mathrm{q}^1}$ is rigorously upper-bounded by $\eta + \varepsilon$. Note that the intermediate approximation $\overline{f}$ should be sharp enough (that is, the approximation degree has to be chosen large enough), so that $\varepsilon \ll \eta$.

4.5 Experimental results

Four examples illustrate our validation method and investigate its limitations, two of which are already treated in [191] from the numerical point of view. First, Airy differential equation exemplifies the polynomial IVP case. Second, the non-polynomial IVP case is illustrated by the mechanical study of the undamped pendulum with variable length. Third, a non-polynomial BVP problem is exemplified by a boundary layer problem. Finally, we apply our method to a practical space mission problem, namely, the trajectory validation in linearized Keplerian dynamics. More detailed applications to space mission problems are exposed in **Chapter 7**, which takes over the article [6].

Remark 4.23 *As explained in Section 4.3.2, the magnitude of the validation parameters N_{val}, h' and d' required by Algorithm* IntopContract *mainly determines the time complexity of the method. In the examples analyzed in this section, we particularly investigate their evolution in function of the parameters of the problems. Usually, they are automatically determined as proposed in Section 4.3.2 (doubling them until the operator $\mathbf{T}$ is proved to be contracting).*

4.5.1 ▶ Airy equation

The Airy function of the first kind is a special function defined by $\text{Ai}(x) = 1/\pi \int_0^{+\infty} \cos(s^3/3 + xs)\text{d}s$ and solution of the Airy differential equation:

$$y''(x) - xy(x) = 0, \tag{4.16}$$

with the initial conditions at 0:

$$\text{Ai}(0) = \frac{1}{3^{2/3}\Gamma(2/3)}, \qquad \text{Ai}'(0) = -\frac{1}{3^{1/3}\Gamma(1/3)}.$$

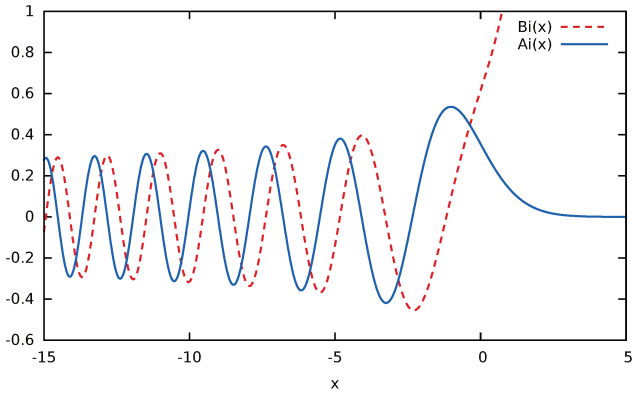
Airy functions, Ai and Bi, depicted in **Figure 4.3**, form together the standard basis of the solutions space of (4.16) (see [2], Chap. 10 *Bessel Functions of Fractional Order*).

In what follows, we apply the validation method on intervals of the form $[-a, 0]$ or $[0, a]$ (for $a > 0$), and investigate its behavior in these two different cases.

Validation over the negative axis

We rigorously approximate Ai over $[-a, 0]$ for some $a > 0$, or equivalently $u(t) = \text{Ai}(-(1+t)a/2)$ over $[-1, 1]$. This appears for instance in quantum mechanics when considering a particle in a one-dimensional uniform electric field. The function u is the solution of the following IVP problem:

$$u''(t) + \frac{a^3}{8}(1+t)u(t) = 0,$$
$$u(-1) = \frac{1}{3^{2/3}\Gamma(2/3)} \quad \text{and} \quad u'(-1) = \frac{a/2}{3^{1/3}\Gamma(1/3)}.$$



■ **Figure 4.3**: Airy functions of the first and second kinds

After the integral transform, we get:

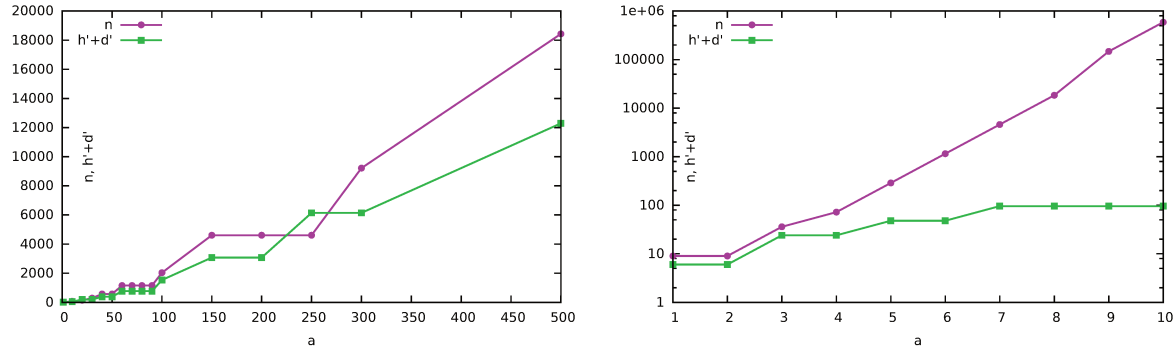
$$k(t,s) = \frac{a^3}{8}(1+t)(t-s), \quad \psi(t) = -\frac{a^3}{8}((1+t)u(-1) + (1+t)^2u'(-1)),$$
$$b_0(t) = \frac{a^3}{16}(T_0(t) + 2T_1(t) + T_2(t)), \quad b_1(t) = -\frac{a^3}{8}(T_0(t) + T_1(t)),$$
$$h = 2, \quad d = 3.$$

Figure 4.4a illustrates the growth of the parameters N_{val}, h' and d' chosen for **Algorithm** IntopContract in order to obtain a contracting Newton-like operator $\mathbf{T}$ when a varies. We observe that N_{val} grows considerably slower than the pessimistic exponential bound claimed in **Section 4.3.2**. In counterpart, the quantity $h' + d'$ is of the order of magnitude of N_{val}, which means that the almost-banded approach computes a dense A and could therefore be replaced by a direct numerical computation of $(\mathbf{1} + \mathbf{K}^{[N_{\text{val}}]})^{-1}$.

As an example, let f° be a degree 48 approximation of Ai over $[-10, 0]$ that reaches machine precision, obtained using the integral reformulation and **Algorithms** OlverTownsendQR and OlverTownsendBackSubs for linear algebra. We want to validate this candidate approximation. The problem is rescaled over $[-1, 1]$ as above, with $a = 10$. Call $\varphi^\circ = f^{\circ\prime\prime}$, and φ^* the exact mathematical solution of the integral equation associated to our problem. With $N_{\text{val}} = 72$, $h' = 24$, $d' = 24$ (automatically obtained as recalled in **Remark 4.23**), **Algorithm** IntopContract produces a contracting operator $\mathbf{T}$ with $\mu = 0.128$. After that, we run **Algorithm** IntopVal: we evaluate $\|\varphi^\circ - \mathbf{T} \cdot \varphi^\circ\|_{\mathrm{Y}^1} = \|\mathbf{A} \cdot (\varphi^\circ + \mathbf{K} \cdot \varphi^\circ - \psi)\|_{\mathrm{Y}^1}$ and obtain $3.87 \cdot 10^{-18}$. So we finally get $\|\varphi^\circ - \varphi^*\|_{\mathrm{Y}^1} \leqslant 3.87 \cdot 10^{-18}/(1 - 0.128) = 4.43 \cdot 10^{-18}$ and $f^\circ = u(-1) + (1+t)u'(-1) + \int_{-1}^{t}\int_{-1}^{s} \varphi^\circ(\tau)\mathrm{d}\tau\mathrm{d}s$ (of degree 50) approximates $u(t) = \mathrm{Ai}(-(1+t)a/2)$ within a Y^1-error equal to $1.78 \cdot 10^{-17}$, which is already beyond machine precision.

■ Validation over the positive axis

We similarly pose $u(t) = \mathrm{Ai}((1+t)a/2)$ to study Ai over the segment $[0, a]$ (with $a > 0$) on the real positive axis. The differential equation and integral transform are similar to the case



(a) over $[-a, 0]$ (b) over $[0, a]$

Figure 4.4: Parameters N_{val} (=: n) and $h' + d'$ chosen during the validation of the Airy function, in function of a

above, except for the signs:

$$u''(t) - \frac{a^3}{8}(1+t)u(t) = 0,$$
$$u(-1) = \frac{1}{3^{2/3}\Gamma(2/3)} \quad \text{and} \quad u'(-1) = -\frac{a/2}{3^{1/3}\Gamma(1/3)},$$
$$k(t,s) = -\frac{a^3}{8}(1+t)(t-s), \quad \psi(t) = \frac{a^3}{8}((1+t)u(-1) + (1+t)^2u'(-1)),$$
$$b_0(t) = -\frac{a^3}{16}(T_0(t) + 2T_1(t) + T_2(t)), \quad b_1(t) = \frac{a^3}{8}(T_0(t) + T_1(t)),$$
$$h = 2, \quad d = 3.$$

Though, we observe a very different behavior for the parameters in **Figure 4.4b**. The truncation order N_{val} grows exponentially and seems in accordance with the pessimistic bound (4.3.2). On the opposite, $h'+d'$ remains significantly smaller than N_{val}, justifying the use of the almost-banded approach.

To understand the difference between the positive and negative cases, let us analyze the behavior of Ai and Bi, see **Figure 4.3**, which is directly linked to the norm of $(\mathbf{1} + \mathbf{K})^{-1}$. Over the negative axis, both Ai and Bi have bounded oscillations. Thus, the intuition is that $\|(\mathbf{1}+\mathbf{K})^{-1}\|_{\text{q1}}$ does not grow so fast when the interval $[-a, 0]$ becomes large, and stays far below the exponential bound $B\exp(2B)$ despite its necessary growth due to these oscillations. On the contrary, Bi grows exponentially fast over the positive axis (we have the asymptotic formula $\text{Bi}(x) \sim \frac{\exp(\frac{2}{3}x^{3/2})}{\sqrt{\pi}x^{1/4}}$ when $x \to +\infty$ **[2,** 10.4.63**]**). This clearly implies that $\|(\mathbf{1} + \mathbf{K})^{-1}\|_{\text{q1}}$ must grow exponentially with a. For the bound based on the Neumann series of the operator, one can reason by analogy with the scalar case, using for example the exponential series $\exp(x) = \sum_{i \geqslant 0} x^i/i!$. When x is a large negative number, the series is alternating, hence, its evaluation $\exp(x)$ is far smaller than the bound $\exp(|x|)$ computed by taking each term of the series in absolute value.

4.5.2 ▶ Undamped pendulum with variable length

Consider the motion of an undamped pendulum with variable length $\ell(t)$, which is modeled by the equation:

$$\theta''(t) + 2\frac{\ell'(t)}{\ell(t)}\theta'(t) + \frac{g}{\ell(t)}\sin\theta(t) = 0, \tag{4.17}$$

where $\theta(t)$ is the angle at time t between the pendulum and its equilibrium position, and $g = 9.81$ the gravitational acceleration. On the time interval $[-1, 1]$ and for a constant variation of the length $\ell(t) = \ell_0(1 + \zeta t)$ (with $|\zeta| < 1$), we analyze the evolution of $\theta(t)$ in a small neighborhood of 0 such that $\sin\theta$ can be linearized into θ. Equation (4.17) becomes:

$$\theta''(t) + \frac{2\zeta}{1+\zeta t}\theta'(t) + \frac{g}{\ell_0(1+\zeta t)}\theta(t) = 0, \qquad \theta(-1) = \theta_0 \ll 1 \text{ and } \theta'(-1) = 0.$$

The coefficients of this equation are not polynomials. Hence, we first provide a Chebyshev model for $\xi(t) = 1/(1 + \zeta t)$ with $|\zeta| < 1$ using the arithmetic operations on RPAs given in **Chapter 3**. **Figure 4.5a** summarizes the obtained error bound ε (for the Y^1-norm) in function of the approximation degree p for different values of ζ, with $\ell_0 = 1$ fixed.

Next, we create and bound the contracting Newton-like operator $\mathbf{T}$. **Figure 4.5b** shows the corresponding values of p (degree of the approximation of $t \mapsto 1/(1+\zeta t)$) and N_{val} (truncation order for the integral operator) we use for **Algorithm** IntopContract, and the values of $h' + d'$, expressing the advantage of taking an almost-banded A instead of a dense one.

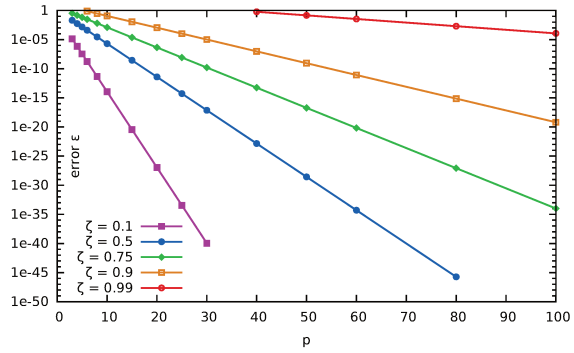
We first observe that N_{val} grows when $|\zeta|$ gets close to 1, which is due to the growth of the Y^1-norm of $t \mapsto 1/(1+\zeta t)$. However, the situation is very different depending on the sign of ζ. When ζ gets close to -1, N_{val} grows exponentially fast. The quantity $h' + d'$ grows more slowly, so that the almost-banded approach helps a little. As for the Airy function, this exponential behavior is due to the large negative coefficient in front of θ' in Equation (4.17). This difficult case corresponds to a decrease in the rope's length, resulting in increasing oscillations of the pendulum (see **Figure 4.5d**). On the contrary, the case $\zeta \to 1$ is easier to treat, since it corresponds to an increase of the rope's length, producing damped oscillations of the pendulum (see **Figure 4.5c**).

The two numerical solutions plotted on **Figures 4.5c** and **4.5d** were certified using **Algorithm** IntopVal. For the damped case ($\ell_0 = 0.1$ and $\zeta = 0.9$), we obtained a Chebyshev model of degree 50 with a Y^1-error equal to $1.40 \cdot 10^{-4}$. The diverging case ($\ell_0 = 0.1$ and $\zeta = -0.9$) used a Chebyshev model of degree 65 with an error of $1.15 \cdot 10^{-4}$.

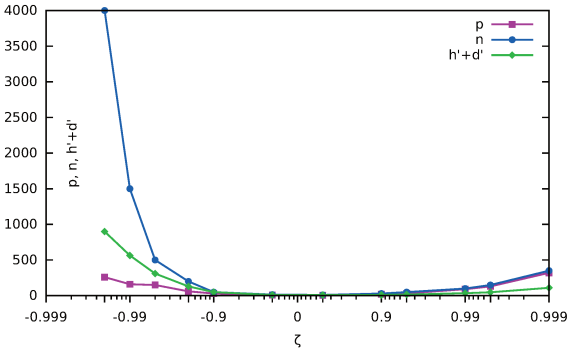
4.5.3 ▶ Boundary layer problem

We take from **[191]** the example of the boundary layer problem, modeled by the following BVP problem, with $\varepsilon > 0$:

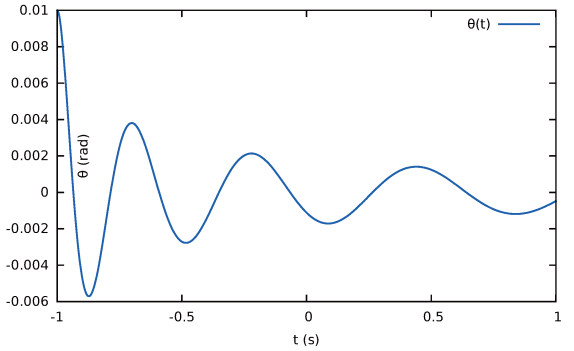
$$\begin{gathered} u''(x) - \frac{2x}{\varepsilon}\left(\cos x - \frac{8}{10}\right)u'(x) + \frac{1}{\varepsilon}\left(\cos x - \frac{8}{10}\right)u(x) = 0, \\ x \in [-1, 1], \quad u(-1) = 1, \quad u(1) = 1. \end{gathered} \tag{4.18}$$



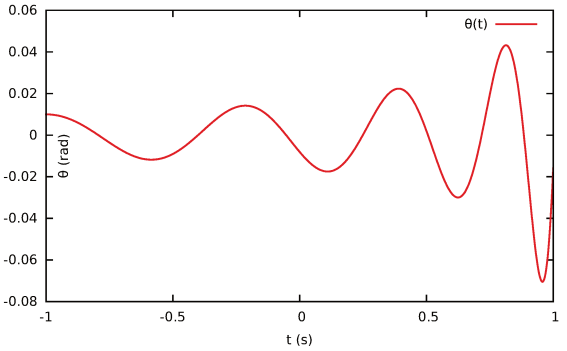
(a) Approximation error ε of the coefficient $t \mapsto 1/(1+\zeta t)$ in function of approximation degree p



(b) Parameters p, $N_{\text{val}}\,(=: n)$ and $h' + d'$ chosen during the validation, for $\ell_0 = 1$ fixed and in function of ζ



(c) Damped oscillations of the pendulum obtained for $\ell_0 = 0.1$ and $\zeta = 0.9$



(d) Diverging oscillations of the pendulum obtained for $\ell_0 = 0.1$ and $\zeta = -0.9$

■ **Figure 4.5**: Validation process for the parametric pendulum

The numerical solution of this BVP is plotted in **Figure 4.6a** for three different values of ε. **Figure 4.6b** shows the basis (u_1, u_2) of the solution space of LODE (4.18) associated to the canonical initial conditions $\{u_1(-1) = 1, u_1'(-1) = 0\}$ and $\{u_2(-1) = 0, u_2'(-1) = 1\}$, for $\varepsilon = 0.001$. Thus, the exact solution u of the BVP is given by:

$$u(x) = u_1(x) + \lambda u_2(x), \quad \text{with } \lambda = \frac{1 - u_1(1)}{u_2(1)}. \tag{4.19}$$

Since $u_1(1)$ and $u_2(1)$ tend to be very large when ε gets close to zero, obtaining u from u_1 and u_2 is an ill-conditioned problem. With $\varepsilon = 0.001$, the obtained approximation using the binary64 (double) format is completely inaccurate (see **Figure 4.6c**). Note that a better solution (regarding the conditioning) is to directly compute the BVP solution with **Algorithms** OlverTownsendQR and OlverTownsendBackSubs as in **[191]**. In any case, validating a candidate solution is useful to detect such numerical troubles.

The first task consists in rigorously approximating the cosine function over $[-1, 1]$. This can be done by a recursive call to our validation method on the differential equation $\xi'' + \xi = 0$ with $\xi(-1) = \cos(-1)$ and $\xi'(-1) = -\sin(-1)$. For this application, a degree 10 Chebyshev model for cos is sufficient.

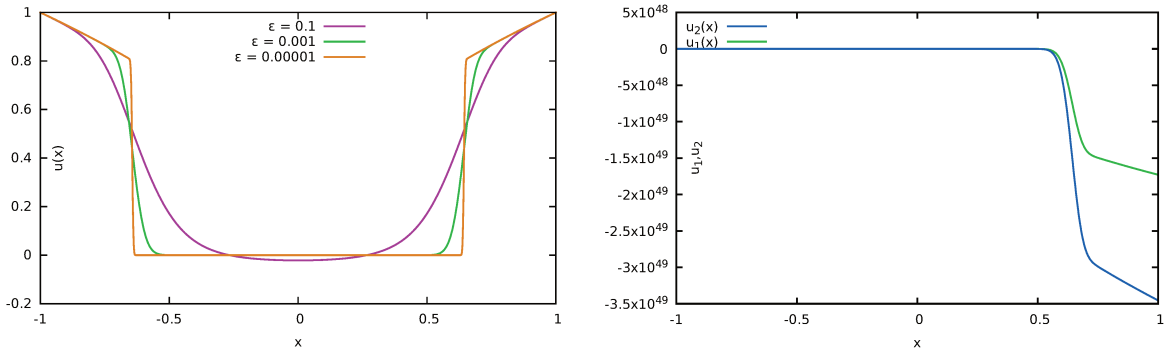
Then, we run IntopContract to get a contracting Newton-like operator. **Figure 4.6d** illustrates the growth of the validation parameters in function of ε. When $\varepsilon > 0$ gets small, the coefficient in front of u' takes large negative values, yielding an exponential growth of $\|(\mathbf{1} + \mathbf{K})^{-1}\|_{\ell^1}$ and hence of the minimal truncation order N_{val} we can choose. Since $h' + d'$ remains small compared to N_{val}, we get here a typical example where the exponential bound prevents us from validating a solution of LODE (4.18) with very small ε, but where however the choice of an almost-banded A allows us to treat intermediate cases: $\varepsilon \in [0.005, 0.01]$.

Next, we compute high-degree Chebyshev models $\mathfrak{u}_1$ and $\mathfrak{u}_2$ for the basis (u_1, u_2). This requires **Algorithms** OlverTownsendQR and OlverTownsendBackSubs to obtain a numerical approximation, and using the previously obtained Newton-like operator $\mathbf{T}$ to certify them with IntopVal. Hence, this step has a linear complexity with respect to the approximation degree we use. Computing the value of λ in Equation (4.19) in interval arithmetics gives a Chebyshev model $\mathfrak{u}$ for the exact solution u using $\mathfrak{u}_1$ and $\mathfrak{u}_2$. Finally, the error associated to the candidate numerical approximate solution u° is obtained by adding the certified error of $\mathfrak{u}$ with the ℓ^1-distance between u° and the polynomial of $\mathfrak{u}$.

As an example, for $\varepsilon = 0.01$, the minimal degree for which we found an approximation of the solution of BVP (4.18) within a certified error of 2^{-53} (corresponding to standard double precision) is 72.

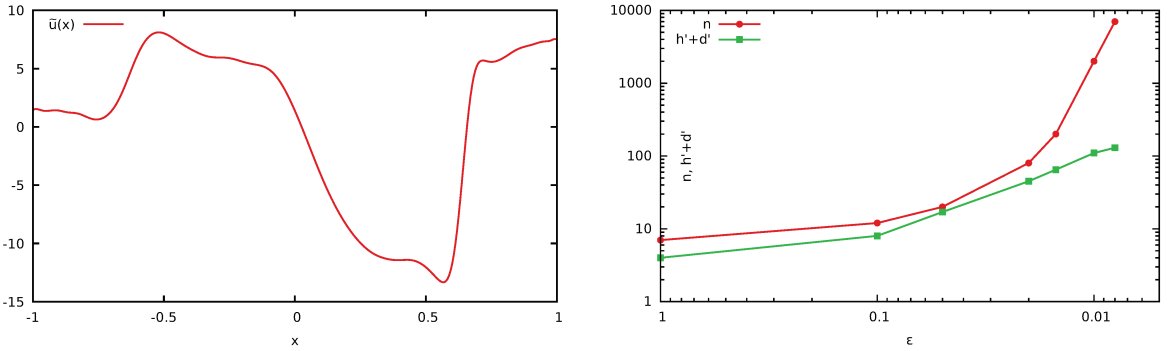
4.5.4 ▶ Spacecraft trajectories using linearized equations for Keplerian motion

We consider the case of Tschauner-Hempel equations, which model the linearized relative motion of an active spacecraft around a passive target (such as the International Space Station for instance) in elliptic orbit around the Earth, provided that their relative distance is small with respect to their distance to the Earth. These equations are very used in robust rendezvous space missions **[245]**, where the accuracy of their computed solutions is at stake. Altough this



(a) Numerical solution of BVP Problem (4.18) for different values of ε

(b) Numerically computed basis (u_1, u_2) of the solution space for $\varepsilon = 0.001$



(c) Inaccurate numerical solution u° obtained with $\varepsilon = 0.001$

(d) Parameters N_{val} (=: n) and $h' + d'$ chosen during the validation, in function of ε

■ **Figure 4.6**: Validation process for the boundary layer problem

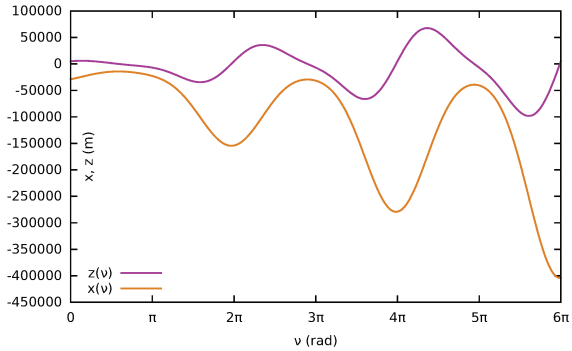
application is further developed in **Chapter 7**, we give a simple example here for illustrative purpose.

Call $e \in [0, 1)$ the eccentricity of the fixed orbit of the target, and let ν be the true anomaly (an angular parameter that defines the position of a body moving along a Keplerian orbit) associated to the target, which is the independent variable in our problem. The in-plane motion of the spacecraft relatively to the target (that is, the component of the motion inside the plane supported by the elliptic orbit of the chaser) is defined using two position variables $x(\nu)$ and $z(\nu)$, satisfying the following linearized system over the interval $[\nu_0, \nu_f]$:

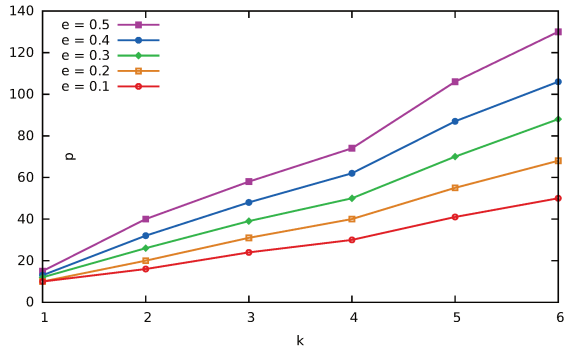
$$z''(\nu) + \left(4 - \frac{3}{1 + e\cos\nu}\right) z(\nu) = c,$$
$$x(\nu) = x(\nu_0) + (x'(\nu_0) - 2z(\nu_0))(\nu - \nu_0) + 2\int_{\nu_0}^{\nu} z(s)\mathrm{d}s,$$
$$c = 4z(\nu_0) - 2x'(\nu_0) \quad \text{and} \quad \nu \in [\nu_0, \nu_f].$$

As an example, fix the eccentricity $e = 0.5$, the interval $[\nu_0, \nu_f] = [0, 6\pi]$ (corresponding to 3 periods) and the initial conditions $(x(\nu_0), z(\nu_0), x'(\nu_0), z'(\nu_0)) = (-3 \cdot 10^4 \text{ m}, 5 \cdot 10^3 \text{ m}, 9 \cdot 10^3 \text{ m} \cdot \text{rad}^{-1}, 4 \cdot 10^3 \text{ m} \cdot \text{rad}^{-1})$. The corresponding functions $x(\nu)$ and $z(\nu)$ are plotted in **Figure 4.7a**. **Figure 4.7c** represents an approximation of degree $N_{\text{app}} = 18$ of $z''(\nu)$ (radial acceleration), together with the rigorous error bound obtained by our method. The dashed curve corresponds to the exact solution, which as expected lies inside the tube defined by our rigorous approximation. One notices that we obtain a quite tight error bound, even for the $\|\cdot\|_\infty$ norm.

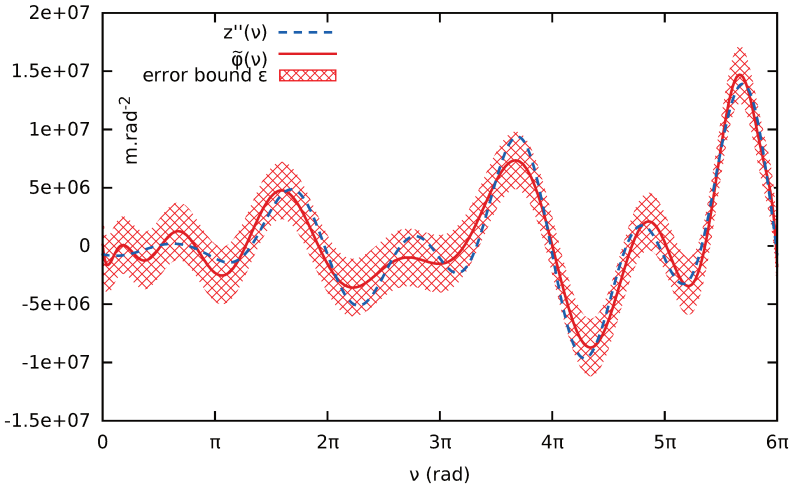
Figure 4.7b gives the minimal degree p corresponding to an approximation of z for which **Algorithm** IntopVal is able to certify an error below one meter, in function of the period length and the eccentricity of the target reference orbit.



(a) Exact representation of $x(\nu)$ and $z(\nu)$ for $\nu \in [0, 6\pi]$ and $e = 0.5$



(b) Approx degree N_{app} (=: p) needed to rigorously approximate z on $[0, k\pi]$ within an error of 1m, in function of k and the eccentricity e



(c) Rigorous approximation φ° of degree $N_{\text{app}} = 18$ of z'', over $[0, 6\pi]$ and for $e = 0.5$

■ **Figure 4.7**: Different validation results related to the spacecraft rendezvous problem

Componentwise Chebyshev Models for Linear Ordinary Differential Systems

5

In the fall of 1972, President Nixon announced that the rate increase of inflation was decreasing. This was the first time a president used the third derivative to advance his case for re-election.

— Hugo E. Rossi

The developments of previous chapters are extended by providing a new framework for a posteriori validation of vector-valued problems with componentwise tight error enclosures, which is applied to solutions of coupled systems of linear ordinary differential equations. More precisely, given a coupled differential system with polynomial or RPA coefficients over a compact interval, and componentwise polynomial approximate solutions in Chebyshev basis, the algorithm outputs componentwise rigorous upper bounds for the approximation errors, with respect to the uniform norm over the interval under consideration.

This work follows closely my conference article "A Newton-like Validation Method for Chebyshev Approximate Solutions of Linear Ordinary Differential Systems" [37], published in 2018, in the proceedings of the *43rd International Symposium on Symbolic and Algebraic Computation* (ISSAC).

Let us firstly introduce some additional notations.

Notations. Let p be a positive integer for the ambient space $\mathbb{R}^p$, whose canonical basis is denoted by $(e_1, \ldots, e_p)$. For a ring $\mathbb{A}$, $\mathcal{M}_p(\mathbb{A})$ denotes the set of order p square matrices, with $\mathbf{1}$ and $\mathbf{0}$ the identity and zero matrices. The order $\leqslant$ over $\mathbb{R}$ is componentwise extended to a (partial) order over $\mathbb{R}^p$ and $\mathcal{M}_p(\mathbb{R})$: for all $u, v \in \mathbb{R}^p$ (resp. A, B in $\mathcal{M}_p(\mathbb{R})$), $u \leqslant v$ if and only if $u_i \leqslant v_i$ for all $i \in [\![1, p]\!]$ (resp. $A \leqslant B$ iff $A_{ij} \leqslant B_{ij}$ for all $i, j \in [\![1, p]\!]$).

Problem statement and contributions. Similarly to the problem considered in **Chapter 4**, we present an *a posteriori* validation algorithm that provides *componentwise* and *tight* error enclosures for Chebyshev approximations to solutions of coupled linear ordinary differential equations (LODEs):

$$Y^{(r)} + A_{r-1}(t)\, Y^{(r-1)} + \cdots + A_1(t)\, Y' + A_0(t)\, Y = G(t), \tag{5.1}$$

of unknown $Y : [-1, 1] \to \mathbb{R}^p$. Coefficients A_i and G must be continuous (vector-valued) functions, given as polynomials with rigorous error bounds. However, for the sake of simplicity, we mainly focus on the polynomial case, and refer to the solutions as *vector-valued D-finite functions* – the general case being deduced as in **Section 4.4.1**. Although such functions can be seen as vectors of (scalar) D-finite functions, the decoupling of the system followed by a possible desingularization step may produce hard to validate scalar LODEs (see **Section 5.3**). Moreover, in the nonpolynomial case, such techniques do not apply.

Using an appropriate integral transform of the linear differential system, we obtain (similarly to what presented in **Proposition 4.5**) a Volterra integral equation of the second kind with polynomial kernel, whence the following problem statement:

Problem 5.1 *For a given integral equation of unknown $\Phi : [-1, 1] \to \mathbb{R}^p$:*

$$\Phi(t) + \int_{-1}^{t} K(t, s)\, \Phi(s) \mathrm{d}s = \Psi(t),$$

with a p-dimensional polynomial kernel $K(t, s) \in \mathcal{M}_p(\mathbb{R}[t, s])$ and $\Psi \in \mathbb{R}[t]^p$, assuming we are given for each component Φ_i^* of the exact solution Φ^* a polynomial approximation Φ_i° in Chebyshev basis, compute componentwise error bounds ε_i, as tight as desired:

$$\|\Phi_i^\circ - \Phi_i^*\|_{\mathrm{q1}} \leqslant \varepsilon_i, \qquad \text{for all } i \in [\![1, p]\!].$$

We recall from **Section 2.2** that $\|\cdot\|_{\mathrm{q1}}$ is a norm for absolutely summable Chebyshev series that upper-bounds the $\|\cdot\|_\infty$ norm over $[-1, 1]$.

Many a posteriori fixed-point validation methods use the Banach fixed-point theorem, following more or less the general framework described in **Section 3.3**. However, in the case we consider, the solution belongs to a product space, and the classical method consisting in endowing it with a global norm fails to produce componentwise tight error enclosures. This is particularly annoying when the components of the system are of different nature (e.g., position and speed) or magnitude.

To overcome this limitation, we consider the notion of *vector-valued* (or *generalized*) metric spaces and *generalized contractions* (or *P-contractions*) **[132, 213, 192]**. The Perov fixed-point theorem **[132, 195]** is a natural extension of the Banach fixed-point theorem and provides componentwise upper bounds for the approximation error. Several works applied this theorem in various settings, for example **[266]** for the Newton method or **[3, 203, 186]** for ODEs with nonlocal conditions. To the best of our knowledge, however, none of these works investigate the existence of lower bounds, nor address validation problems. Based on a new refinement with lower bounds for the Perov fixed-point theorem, we propose a validation algorithm to solve **Problem 5.1**.

Outline. **Section 5.1** introduces a general framework for componentwise fixed-point validation in generalized metric spaces. In **Section 5.2**, we design a Newton-like validation algorithm for Chebyshev approximations of vector-valued D-finite functions. After that, **Section 5.3** details the validation of a two-dimensional highly oscillating system. For completeness, we also provide a comparison with a decoupling technique that boils down to solving scalar LODEs. Finally, concluding remarks about the methods developed in **Chapters 4** and **5** are proposed in **Section 5.4**, together with future directions that deserve further investigations.

5.1 A framework for vector-valued validation problems

We address the general problem of componentwise validating an approximation x° to the exact solution x^* of a fixed-point equation $x = \mathbf{T} \cdot x$. **Section 5.1.1** gives a rigorous definition of "several components and norms" with the notion of generalized metric spaces, leading to the Perov fixed-point theorem. **Section 5.1.2** presents a new result that complements the Perov theorem with lower bounds on the componentwise approximation errors.

A toy example in the plane

Throughout this section, the following toy example will be used to illustrate the vector-valued validation framework. Consider the trigonometric equation $\sin^3\vartheta + \cos 3\vartheta = 0$ for $\vartheta \in \mathbb{R}$. By introducing $c = \cos x$ and $s = \sin x$, this is equivalent to finding the roots of the following polynomial system in the plane (c, s):

$$\mathbf{F} \cdot (c, s) = \begin{pmatrix} s^3 + 4c^3 - 3c \\ c^2 + s^2 - 1 \end{pmatrix} = 0.$$

Let $x^* = (c^*, s^*)$ be an exact solution and $x^\circ := (c^\circ, s^\circ) = (0.84, 0.55)$ an approximation of it. In order to validate this solution with respect to a given norm $\|\cdot\|$ on $\mathbb{R}^2$, we define a Newton-like operator $\mathbf{T} \cdot (c, s) := (c, s) - \mathbf{A} \cdot \mathbf{F} \cdot (c, s)$ with $\mathbf{A} := \begin{pmatrix} 0.25 & -0.20 \\ -0.37 & 1.2 \end{pmatrix} \approx (\mathcal{D}\mathbf{F}_{x^\circ})^{-1} \in \mathcal{M}_2(\mathbb{R})$ an approximate inverse of the Fréchet derivative $\mathcal{D}\mathbf{F}_{x^\circ}$ of $\mathbf{F}$ at x°. Since $\mathbf{A}$ is injective, its fixed points are exactly the roots of $\mathbf{F}$. In this example, $\mathbf{F}$ is nonlinear, so one must find a *strongly stable ball* over which $\mathbf{T}$ is contracting, for the Banach Theorem **3.10** to apply, that is, determine a radius $r > 0$ satisfying the following two conditions:

(i) $\lambda := \sup_{\|x - x^\circ\| \leqslant r} \|\mathbf{1} - \mathbf{A} \cdot \mathcal{D}\mathbf{F}_x\| < 1$;

(ii) $\|x^\circ - \mathbf{T} \cdot x^\circ\| + \lambda r \leqslant r$.

If such a radius exists, then by the Banach fixed-point theorem, we have $\|x^\circ - x^*\| \leqslant \frac{\|\mathbf{A} \cdot \mathbf{F} \cdot x\|}{(1-\lambda)}$. However, such a bound captures a "global" error, which may not be what we expect, if, for example, the two components are of different nature (e.g., position and velocity), or differ by several orders of magnitude.

5.1.1 ▶ Generalized Metric Spaces and Perov Fixed-Point Theorem

Definition 5.2 *Let X be a set (resp. E a linear space). A function $d : X \times X \to \mathbb{R}^p_+$ (resp. $\|\cdot\| : E \to \mathbb{R}^p_+$) is a* vector-valued *or* generalized *metric (resp. norm) if for all x, y, z in X or E and $\lambda \in \mathbb{R}$:*

- $d(x,y) = 0$ *iff* $x = y$, *resp.* $\|x\| = 0$ *iff* $x = 0$;
- $d(x,y) = d(y,x)$, *resp.* $\|\lambda x\| = |\lambda|\|x\|$;
- $d(x,y) \leqslant d(x,z) + d(z,y)$, *resp.* $\|x+y\| \leqslant \|x\| + \|y\|$.

Then (X,d) *(resp.* $(E, \|\cdot\|)$*) is a* vector-valued *or* generalized *metric space (resp. linear space).*

A straightforward example is the product of p metric spaces (X_i, d_i), $i \in [\![1,p]\!]$ (resp. p normed linear spaces $(E, \|\cdot\|_i)$) and the vector-valued metric $d(x,y) = (d_1(x_1,y_1), \dots, d_p(x_p,y_p))$ (resp. the vector-valued norm $\|x\| = (\|x_1\|_1, \dots, \|x_p\|_p)$).

Remark 5.3 *A vector-valued metric space (respectively a vector-valued normed linear space) can be trivially seen as a metric space (respectively a normed linear space) by taking the maximum of all the components of the vector-valued metric (respectively norm). We therefore recover all the useful topological notions of convergence, limit, neighborhood, completeness, etc.*

In the context of vector-valued metric spaces, the notion of contracting map needs to be generalized. Let $\mathcal{M}_p^{\to 0}(\mathbb{R}) \subseteq \mathcal{M}_p(\mathbb{R})$ denote the *convergent to zero matrices*, that is the matrices M such that $M^k \to 0$ as $k \to \infty$. Equivalently, these are matrices M with spectral radius $\rho(M) < 1$. Then, $\mathcal{M}_p^{\to 0}(\mathbb{R}_+) = \mathcal{M}_p^{\to 0}(\mathbb{R}) \cap \mathcal{M}_p(\mathbb{R}_+)$ denotes those among them with nonnegative coefficients.

Definition 5.4 *Let* (X,d) *be a vector-valued metric space and* $\mathbf{T} : X \to X$ *an operator.*

- $\mathbf{T}$ *is* Λ*-Lipschitz for some* $\Lambda \in \mathcal{M}_p(\mathbb{R}_+)$ *if:*

$$d(\mathbf{T}\cdot x, \mathbf{T}\cdot y) \leqslant \Lambda\, d(x,y), \qquad \text{for all } x,y \in X.$$

- *If moreover* Λ *is convergent to* 0 *(*$\Lambda \in \mathcal{M}_p^{\to 0}(\mathbb{R}_+)$*), then* $\mathbf{T}$ *is said to be a generalized contraction.*

Using these definitions, the Perov fixed-point theorem[1] is a generalization of the Banach fixed-point theorem.

Theorem 5.5 (Perov) *Let* (X,d) *be a complete vector-valued metric space and* $\mathbf{T} : X \to X$ *a generalized contraction with a Lipschitz matrix* $\Lambda \in \mathcal{M}_p^{\to 0}(\mathbb{R}_+)$*. Then:*

(5.5 i) $\mathbf{T}$ *admits a unique fixed-point* $x^* \in X$*;*

(5.5 ii) *For every* $x^\circ \in X$*, the iterated sequence defined by* $x_0 = x^\circ$ *and* $x_{n+1} = \mathbf{T}\cdot x_n$ *converges to* x^* *with the following upper bound on the approximation error:*

$$d(x_n, x^*) \leqslant \Lambda^n \left(\mathbf{1} - \Lambda\right)^{-1} d(x^\circ, \mathbf{T}\cdot x^\circ), \qquad \text{for all } n \in \mathbb{N}. \tag{5.2}$$

A reference proof may be found in [192], but we give below the main ideas, which are useful for what follows.

[1] Although commonly attributed to Perov [195] (in Russian), the idea of generalizing the Banach fixed-point theorem to generalized norms for investigating the componentwise errors in an iterative process first appeared in Kantorovich's work [132] (in Russian).

Proof. (5.5 i) Endow X with the metric $d_\infty(x,y) = \|d(x,y)\|_\infty = \max_{1\leqslant i\leqslant p} d_i(x)$, so that (X, d_∞) is a complete metric space. For an order p square matrix A, define:

$$\|A\|_\infty := \max_{1\leqslant i\leqslant p} \sum_{1\leqslant j\leqslant p} |A_{ij}| = \sup_{\|x\|_\infty\leqslant 1} \|A\,x\|_\infty.$$

Since $\Lambda^k \to 0$, there is a k such that $\mu = \|\Lambda^k\|_\infty < 1$. Then $\mathbf{T}^k$ is a μ-contraction for the d_∞ metric, so that the Banach theorem applies and gives x^* as the unique fixed-point of $\mathbf{T}^k$. Hence $\mathbf{T}$ can have at most one fixed point. From the following inequality:

$$d_\infty(x^*, \mathbf{T}\cdot x^*) = d_\infty(\mathbf{T}^k\cdot x^*, \mathbf{T}^{k+1}\cdot x^*) \leqslant \|\Lambda^k\|_\infty d_\infty(x^*, \mathbf{T}\cdot x^*) < d_\infty(x^*, \mathbf{T}\cdot x^*),$$

we get that $x^* = \mathbf{T}\cdot x^*$ is the unique fixed point of $\mathbf{T}$.

(5.5 ii) Let $x^\circ \in X$. Since $d(x^\circ, x^*) \leqslant d(x^\circ, \mathbf{T}\cdot x^\circ) + d(\mathbf{T}\cdot x^\circ, x^*) \leqslant d(x^\circ, \mathbf{T}\cdot x^\circ) + \Lambda\, d(x^\circ, x^*)$, we get:

$$(\mathbf{1} - \Lambda)\, d(x^\circ, x^*) \leqslant d(x^\circ, \mathbf{T}\cdot x^\circ). \tag{5.3}$$

Since $\Lambda^k \to 0$ as $k \to \infty$, it is easy to prove that $\mathbf{1} - \Lambda$ is nonsingular, with nonnegative inverse $(\mathbf{1} - \Lambda)^{-1} = \sum_{k\geqslant 0} \Lambda^k \geqslant \mathbf{0}$. Therefore, multiplying both members of Inequality (5.3) by $(\mathbf{1} - \Lambda)^{-1}$ is licit, so as to obtain the upper bound (5.2) for $n = 0$. The general bound for $n \geqslant 0$ follows from the fact that $\mathbf{T}$ is Λ-Lipschitz. □

Perov theorem applied to the toy example

Endowing $\mathbb{R}^2$ with the vector-valued norm $\|(c,s)\| := (|c|, |s|)$ does not change the definition of $\mathbf{T}$. The two conditions needed to apply the Banach fixed-point theorem are adapted to the Perov theorem as follows. Choose a *multi-radius* $r = (r_1, r_2)$ such that

(i) $\Lambda := \left(\sup_{\|x-x^\circ\|\leqslant r} |(\mathcal{D}\mathbf{T}_x)_{ij}|\right)_{1\leqslant i,j\leqslant 2}$ satisfies $\rho(\Lambda) < 1$;

(ii) $\|x^\circ - \mathbf{T}\cdot x^\circ\| + \Lambda\, r \leqslant r$.

For $r = (0.005, 0.005)$, one obtains:

$$\Lambda = \begin{pmatrix} 5.81 & 1.31 \\ 5.63 & 3.40 \end{pmatrix} \cdot 10^{-2}, \qquad \rho(\Lambda) = 7.57 \cdot 10^{-2},$$

which satisfies (i) and (ii). Hence, **Theorem 5.5** gives:

$$|c^\circ - c^*| \leqslant 2.90 \cdot 10^{-3}, \qquad |s^\circ - s^*| \leqslant 3.65 \cdot 10^{-3}.$$

To assess the tightness of these bounds, the lower bounds given in next section will be useful.

5.1.2 ▶ Lower Bounds and Error Enclosures

Let $\varepsilon = d(x^\circ, x^*) \in \mathbb{R}^p_+$ be the vector of unknown errors and $\eta = d(x^\circ, \mathbf{T}\cdot x^\circ) \in \mathbb{R}^p_+$. By the triangle inequality, ε is circumscribed into a polytope of $\mathbb{R}^p_+$:

$$\begin{aligned} (\mathbf{1} - \Lambda)\,\varepsilon &\leqslant \eta, \\ (\mathbf{1} + \Lambda)\,\varepsilon &\geqslant \eta, \\ \varepsilon &\geqslant 0. \end{aligned} \tag{5.4}$$

The first inequality gives the upper bounds $\varepsilon^+ = (\mathbf{1} - \Lambda)^{-1}\,\eta$, as stated by **Theorem 5.5** (with $n = 0$). However, the second one does not directly give the desired lower bounds, say ε^-, because the inverse $(\mathbf{1} + \Lambda)^{-1} = \sum_{k\geqslant 0}(-\Lambda)^k$ is not nonnegative in general. It is clear that each ε_i^- is given by the i-th coordinate of some vertex of this polytope. Instead of testing its 2^p vertices, **Theorem 5.6** identifies the correct one.

Theorem 5.6 (Lower bounds for the Perov theorem) *With the above notations, for each $i \in [\![1, p]\!]$, the lower bound ε_i^- on the i-th component ε_i of the approximation error of x° to x^* is given by the i-th component of the vertex defined by the intersection of the i-th lower-bound constraint together with all the j-th upper-bound constraints with $j \neq i$ from (5.4). Formally:*

$$\varepsilon_i \geqslant \varepsilon_i^- \qquad \text{with} \qquad \varepsilon_i^- = e_i^T\,(\mathbf{1} - D_i\,\Lambda)^{-1}\,\eta,$$

where D_i is the order p diagonal matrix defined by $(D_i)_{ii} = -1$ and $(D_i)_{jj} = 1$ for $j \neq i$.

Before proving this theorem, we give two technical lemmas.

Lemma 5.7 *Let $A \in \mathcal{M}_p^{\to 0}(\mathbb{R}_+)$ be a convergent to zero nonnegative matrix and $B \in \mathcal{M}_p(\mathbb{R})$ be a matrix whose entries are dominated by those of A:*

$$|B_{ij}| \leqslant A_{ij}, \qquad \text{for all } i, j \in [\![1, p]\!].$$

Then B is convergent to zero.

Proof. Since A has nonnegative entries which bound those of B, it can be easily shown by the triangle inequality that for any exponent $k \geqslant 0$, $|B_{ij}^k| \leqslant A_{ij}^k$ for all $i, j \in [\![1, p]\!]$. This directly implies the conclusion of **Lemma 5.7**. □

Lemma 5.8 *Let $\Lambda \in \mathcal{M}_p^{\to 0}(\mathbb{R}_+)$ be a convergent to zero nonnegative matrix. Then, for every $i \in [\![1, p]\!]$, $\Lambda - D_i$ is nonsingular and the entries on the i-th row of its inverse are nonnegative.*

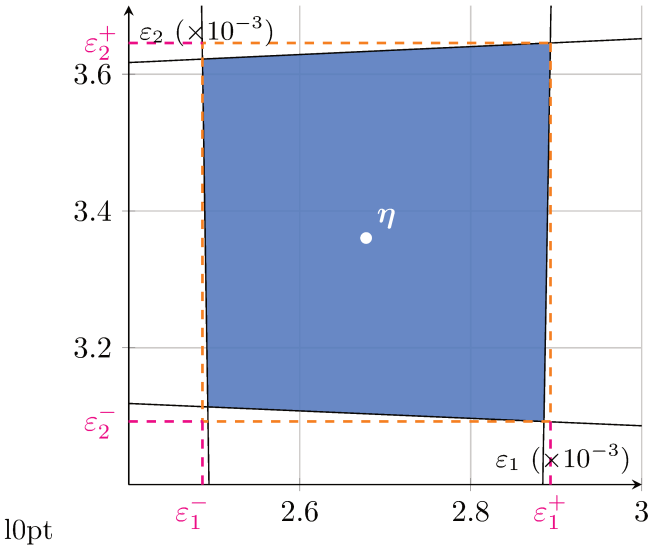
Proof of Lemma 5.8. First, $\mathbf{1} - D_i\,\Lambda$ is nonsingular because $D_i\,\Lambda$ is convergent to zero by use of **Lemma 5.7**, since its entries are clearly dominated by those of $\Lambda \in \mathcal{M}_p^{\to 0}(\mathbb{R}_+)$. Hence so is $\Lambda - D_i$.

Then we prove that $\mathbf{1} - \Lambda$ and $\mathbf{1} - D_i\,\Lambda$ both have positive determinant. The segment $\mathbf{1} - \tau\Lambda$ ($\tau \in [0, 1]$) connects $\mathbf{1}$ to $\mathbf{1} - \Lambda$, and all these matrices are nonsingular, because $\tau\Lambda$ converges to zero according to **Lemma 5.7**. Since $\det(\mathbf{1}) = 1 > 0$, we get by connectedness that $\det(\mathbf{1} - \Lambda) > 0$. A similar argument proves that $\det(\mathbf{1} - D_i\,\Lambda) > 0$, and hence $\det(D_i - \Lambda) < 0$.

Remember that for a nonsinglular matrix M, we have $M^{-1} = (\det M)^{-1} Cof(M)^T$, where $Cof(M)$ is the cofactor matrix of M, whose entries are the minors of M. Noticing that $Cof(D_i - \Lambda)_{ji} = Cof(\mathbf{1} - \Lambda)_{ji}$ for $j \in [\![1, p]\!]$ and using the fact that $\det(D_i - \Lambda) < 0$, $\det(\mathbf{1} - \Lambda) > 0$ and all entries in $(\mathbf{1} - \Lambda)^{-1}$ are nonnegative, we conclude that all entries on the i-th row of $(D_i - \Lambda)^{-1}$ are non-positive. □

Proof of Theorem 5.6. Among the Inequalities (5.4), take the p upper-bound constraints and replace the i-th one by the corresponding lower-bound constraint. Multiply these $p - 1$ upper-bound constraints by -1 to obtain the following system of inequalities:

$$(\Lambda - D_i)\,\varepsilon \geqslant -D_i\,\eta. \tag{5.5}$$



l0pt

Figure 5.1: Error polytope for the toy example.

From **Lemma 5.8**, $\Lambda - D_i$ is nonsingular and its inverse has nonnegative coefficients on its i-th row. Hence we can multiply (5.5) by $(\Lambda - D_i)^{-1}$ and only keep the resulting i-th constraint:

$$\varepsilon_i = e_i^T \, (\Lambda - D_i)^{-1} \, (\Lambda - D_i) \, \varepsilon \geqslant e_i^T \, (\Lambda - D_i)^{-1} \, (-D_i) \cdot \eta = e_i^T \, (\mathbf{1} - D_i \, \Lambda)^{-1} \, \eta.$$

□

Remark 5.9 *Contrary to the one-dimensional case, ε_i^- may be negative (we then round it to 0): the overestimation factor of ε_i^+ provided by **Theorem 5.5** is not controlled. A tighter enclosure can be obtained with a more contracting* $\mathbf{T}$ *(see **Section 5.1.3**).*

Lower bounds for the toy example

The polytope given by the linear constraints (5.4) is depicted in **Figure 5.1**. The top right vertex corresponds to $(\varepsilon_1^+, \varepsilon_2^+)$. Also, the ε_1^- (resp. ε_2^-) is given by the top left (resp. bottom right) vertex, which is consistent with **Theorem 5.6**. This gives the following numerical enclosures:

$$\begin{aligned} \varepsilon_1^- = 2.48 \cdot 10^{-3} \leqslant \varepsilon_1 = |c^\circ - c^*| \leqslant \varepsilon_1^+ = 2.90 \cdot 10^{-3}, \\ \varepsilon_2^- = 3.09 \cdot 10^{-3} \leqslant \varepsilon_2 = |s^\circ - s^*| \leqslant \varepsilon_2^+ = 3.65 \cdot 10^{-3}. \end{aligned} \tag{5.6}$$

5.1.3 ▶ Tightness of Error Enclosures

In the one-dimensional case with a contracting operator of Lipschitz constant $\lambda \in (0,1)$, the ratio of the upper bound and the lower bound given by the Banach fixed-point theorem is equal to $(1+\lambda)/(1-\lambda) > 1$. This quantity does not depend on the approximation x°, and uniformly

tends to 1 as $\lambda \to 0$, justifying the principle: the more contracting the operator is, the tighter the obtained enclosure is.

This section aims at extending this study to the vectorial case, for the bounds obtained from **Theorems 5.5** and **5.6**. The following lemma quantifies how much the obtained enclosure $[\varepsilon_i^-, \varepsilon_i^+]$ of $\varepsilon_i = d(x^\circ, x^*)_i$ may overapproximate it.

Lemma 5.10 *Let* $\mathbf{T}$ *be contracting of Lipschitz matrix* $\Lambda \in \mathcal{M}_p^{\to 0}(\mathbb{R}_+)$, x^* *its unique fixed-point,* x° *an approximation,* $\varepsilon = d(x^\circ, x^*)$ *the approximation error, and* $\eta = d(x^\circ, \mathbf{T} \cdot x^\circ)$. *Then for all* $i \in [\![1, p]\!]$*:*

$$\begin{aligned} \varepsilon_i^+ &:= e_i^T (1-\Lambda)^{-1} \eta \leqslant e_i^T (1-\Lambda)^{-1} (1+\Lambda) \varepsilon, \\ \varepsilon_i^- &:= e_i^T (1-D_i\Lambda)^{-1} \eta \geqslant e_i^T (1-D_i\Lambda)^{-1} (1+D_i\Lambda) \varepsilon. \end{aligned} \tag{5.7}$$

The overapproximation ratio can be bounded as follows:

$$\frac{\varepsilon_i^+}{\varepsilon_i^-} \leqslant \frac{1+\alpha\nu_i}{1-\alpha\nu_i}, \quad \text{with } \nu_i = \frac{\|\varepsilon\|_\infty}{\varepsilon_i} \text{ and } \alpha = \frac{2\|\Lambda\|_\infty}{1-\|\Lambda\|_\infty}, \tag{5.8}$$

provided that $\alpha\nu_i < 1$, *where* $\|\varepsilon\|_\infty := \max\limits_{1 \leqslant i \leqslant p} |\varepsilon_i|$ *is the infinity norm over* $\mathbb{R}^p$ *and* $\|\Lambda\|_\infty := \max\limits_{1 \leqslant i \leqslant p} \sum_{j=1}^p |\Lambda_{ij}|$ *the associated operator norm.*

In particular, **Lemma 5.10** shows that the ratio now depends not only on Λ, but also on $\varepsilon = d(x^\circ, x^*)$.

Proof. For the first inequality, we have:

$$\varepsilon_i^+ = e_i^T (1-\Lambda)^{-1} \eta \leqslant e_i^T (1-\Lambda)^{-1} (1+\Lambda) \varepsilon,$$

since $\eta \leqslant (1+\Lambda)\varepsilon$ by Equation **(5.4)** and $(1-\Lambda)^{-1}$ has nonnegative coefficients.

For the second inequality, consider $\overline{\eta} := (1+D_i\Lambda)\varepsilon$, so that:

$$\begin{aligned} \overline{\eta}_i &= e_i^T (1-\Lambda)\varepsilon \leqslant \eta_i, \\ \overline{\eta}_j &= e_j^T (1+\Lambda)\varepsilon \geqslant \eta_j, \text{ for } j \neq i \end{aligned}$$

Since by **Lemma 5.8**, $e_i^T (1-D_i\Lambda)^{-1}$ (the i-th line of $(1-D_i\Lambda)^{-1}$) has a positive entry on the i-th position and nonpositive entries on the other ones, we have:

$$\varepsilon_i^- = e_i^T (1-D_i\Lambda)^{-1} \eta \geqslant e_i^T (1-D_i\Lambda) \overline{\eta}.$$

In order to prove **(5.8)**, we notice that:

$$e_i^T (1-\Lambda)^{-1} (1+\Lambda)\varepsilon = \varepsilon_i + 2e_i^T \sum_{n \geqslant 1} \Lambda^n \varepsilon \geqslant \varepsilon_i + 2\sum_{n \geqslant 1} \|\Lambda\|_\infty \|\varepsilon\|_\infty = \varepsilon_i + \alpha\|\varepsilon\|_\infty,$$

under the condition that $\|\Lambda\|_\infty < 1$, which is automatic from assumption $\alpha\nu_i < 1$. The inequality:

$$e_i^T (1-D_i\Lambda)^{-1} (1+D_i\Lambda)\varepsilon \geqslant \varepsilon_i - \alpha\|\varepsilon\|_\infty,$$

is proven similarly by considering $D_i\Lambda$ instead of Λ and $\|D_i\Lambda\|_\infty = \|\Lambda\|_\infty$. Taking the quotient of these two inequalities yields the bound **(5.8)**. □

Tightness cones

For a fixed Λ, the overapproximation ratio depends on the error distribution of the ε_i. More specifically, using Inequality (5.7) of **Lemma 5.10**, the constraint $\varepsilon_i^+/\varepsilon_i^- \leqslant \kappa$ for some ratio $\kappa > 1$ is fulfilled when:

$$e_i^T (1-\Lambda)^{-1}(1+\Lambda)\,\varepsilon \leqslant \kappa\, e_i^T (1-D_i\,\Lambda)^{-1}(1+D_i\,\Lambda)\,\varepsilon.$$

The intersection of these constraints for $i \in [\![1,p]\!]$ defines a cone $\mathcal{C}_\kappa$ in $\mathbb{R}_+^p$ of points ε for which the computed overapproximation ratio does not exceed κ. Under a certain value for κ, $\mathcal{C}_\kappa$ is empty, meaning that $\mathbf{T}$ is not contracting enough to achieve this ratio, whatever ε is. The cone $\mathcal{C}_\kappa$ grows to a *limit cone* $\mathcal{C}_\infty$ as $\kappa \to +\infty$: a point outside $\mathcal{C}_\infty$ means that the componentwise error distribution is so unbalanced that some lower bound ε_i^- is negative (hence rounded to zero). **Figure 5.2** illustrates the cones $\mathcal{C}_\kappa$ for different values of κ and the limit cone $\mathcal{C}_\infty$ arising in our toy example.

Validation up to a given ratio

When the problem consists in finding an error enclosure $[\varepsilon_i^-, \varepsilon_i^+]$ for a given approximation x° of x^* such that $\varepsilon_i^+/\varepsilon_i^- \leqslant \kappa$, the fixed-point validation operator $\mathbf{T}$ must be chosen sufficiently contracting. The following lemma characterizes how small the Lipschitz matrix Λ must be in that case.

Lemma 5.11 *Let $\mathbf{T}$ be contracting of Lipschitz matrix $\Lambda \in \mathcal{M}_p^{\to 0}(\mathbb{R}_+)$, x° an approximation of its unique fixed point x^*, $\varepsilon = d(x^\circ, x^*)$ the approximation error, and $\kappa > 1$. If*

$$\|\Lambda\|_\infty \leqslant \frac{\beta}{2+\beta} \quad \textit{with } \beta = \frac{1}{\nu_i}\frac{\kappa-1}{\kappa+1} \textit{ and } \nu_i = \frac{\|\varepsilon\|_\infty}{\varepsilon_i},$$

*then the upper and lower bounds computed using **Theorems 5.5** and **5.6** satisfy $\varepsilon_i^+/\varepsilon_i^- \leqslant \kappa$.*

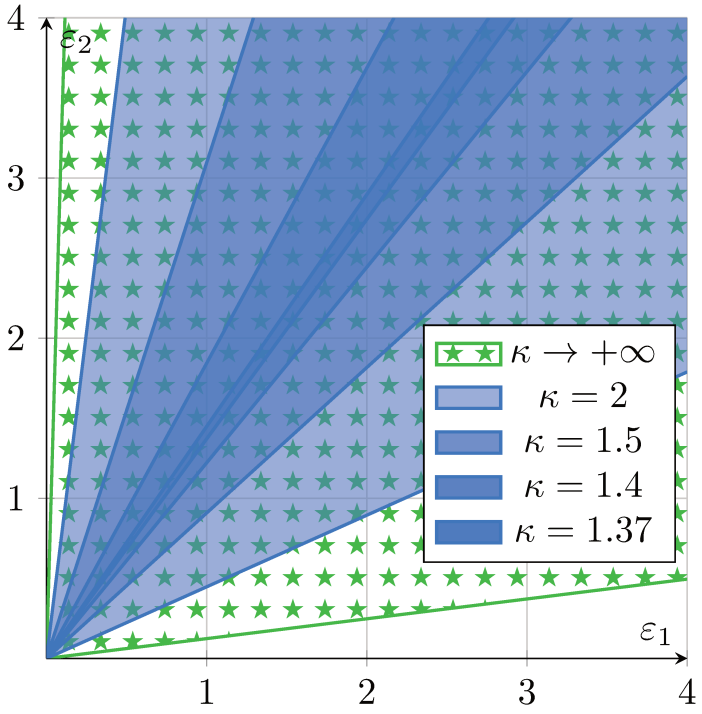
Proof. By **Lemma 5.10**, having:

$$\frac{1+\alpha\nu_i}{1-\alpha\nu_i} \leqslant \kappa, \quad \text{with } \alpha = \frac{2\|\Lambda\|_\infty}{1-\|\Lambda\|_\infty},$$

is sufficient to ensure $\varepsilon_i^+/\varepsilon_i^- \leqslant \kappa$ (note that the assumption $\|\Lambda\|_\infty \leqslant \beta/(2+\beta)$ implies $\alpha\nu_i < 1$). This in turn is equivalent to $\alpha \leqslant \beta$ and finally $\|\Lambda\|_\infty \leqslant \beta/(2+\beta)$. □

5.2 Componentwise validation of Chebyshev approximations

The validation procedure for vector-valued D-finite functions closely follows the scalar case described in **Chapter 4**. After a short reminder on integral transforms for systems of LODEs



■ **Figure 5.2**: Tightness cones for the toy example

in **Section 5.2.1**, we explain in **Section 5.2.2** how to exploit the almost-banded structure of the resulting linear problem on the Chebyshev coefficients for an efficient numerical solving. Finally, **Section 5.2.3** presents the validation step in this vectorial setting, together with complexity estimates.

5.2.1 ▶ D-finite equations and integral transforms

We consider a generic p-dimensional order r system of LODEs over the compact interval $[-1, 1]$:

$$Y^{(r)} + A_{r-1}(t)\, Y^{(r-1)} + \cdots + A_1(t)\, Y' + A_0(t)\, Y = G(t),$$

of unknown $Y = (Y_1, \ldots, Y_p) : [-1, 1] \to \mathbb{R}^p$, with polynomial coefficients $A_k = (a_{kij})_{1 \leqslant i,j \leqslant p} \in \mathcal{M}_p(\mathbb{R}[t])$ and right-hand side $G = (G_1, \ldots, G_p) \in \mathbb{R}[t]^p$. We also fix initial conditions at -1:

$$Y^{(i)}(-1) = v_i, \qquad v_i \in \mathbb{R}^p, \quad \text{for all } i \in [\![0, r-1]\!]. \tag{5.9}$$

Together, (5.1) and (5.9) form an *Initial Value Problem* (IVP).

As mentioned in the previous chapter, several barriers arise when working directly on a differential equation (5.1). Similarly to the scalar case, a common way to circumvent these limitations is to apply an integral transform onto the IVP problem so as to obtain an equivalent *Volterra integral equation of the second kind* over $[-1, 1]$:

$$\Phi + \mathbf{K} \cdot \Phi = \Psi, \quad \text{with} \quad \mathbf{K} \cdot \Phi(t) := \int_{-1}^{t} K(t, s)\, \Phi(s) \mathrm{d}s, \tag{5.10}$$

with a bivariate polynomial kernel $K = (k_{ij})_{1\leqslant i,j\leqslant p} \in \mathcal{M}_p(\mathbb{R}[t,s])$ and right-hand side $\Psi = (\Psi_1, \dots, \Psi_p) \in \mathbb{R}[t]^p$. Depending on the integral transform, the unknown function $\Phi = (\Phi_1, \dots, \Phi_p) : [-1,1] \to \mathbb{R}^p$ can be either Y or one of its derivatives. For example, [19] acts over Y, whereas the integral transform in previous chapter considers the last derivative $Y^{(r)}$.

In any case, $\mathbf{K} : \Phi \mapsto \int_{-1}^{t} K(t,s)\,\Phi(s)\mathrm{d}s$ is a bounded linear operator from $(\mathrm{Y}^1)^p$ to itself. We may describe it by blocks $\mathbf{K} = (\mathbf{K}_{ij})_{1\leqslant i,j\leqslant p}$, where each $\mathbf{K}_{ij}$ is a one-dimensional integral operator of kernel $k_{ij}(t,s)$, obtained for example as in **Proposition 4.5**. By decomposing $k_{ij}(t,s)$ in Chebyshev basis with respect to s, we obtain unique polynomials $b_{ijk}(t)$ such that

$$k_{ij}(t,s) = \sum_{k=0}^{\kappa_{ij}} b_{ijk}(t) T_k(s), \quad \mathbf{K}_{ij} \cdot \varphi(t) = \sum_{k=0}^{\kappa_{ij}} b_{ijk}(t) \int_{-1}^{t} T_k(s)\varphi(s)\mathrm{d}s.$$

Recalling properties of integral operators from **Section 4.1**, the (infinite dimensional) matrix representation of $\mathbf{K}_{ij} : \mathrm{Y}^1 \to \mathrm{Y}^1$ has a so-called (h_{ij}, d_{ij}) *almost-banded* structure [191], meaning that the nonzero entries are located on the h_{ij} first rows (*horizontal band* with *initial entries*) and the diagonal plus the first d_{ij} upper and lower diagonals (*diagonal band* with *diagonal entries*), with $h_{ij} = \max\limits_{0\leqslant k\leqslant \kappa_{ij}} \deg b_{ijk}(t)$ and $d_{ij} = 1 + \deg k_{ij}(t,s) = 1 + \max\limits_{0\leqslant k\leqslant \kappa_{ij}} (k + \deg b_{ijk}(t))$.

5.2.2 ▶ Efficient numerical solving

The integral equation (5.10) is an infinite-dimensional linear system over the Chebyshev coefficients of the unknown function Φ. The *projection method* (also sometimes called *Galerkin method* [96]) consists in truncating for a given index N_{app} and solving the obtained finite-dimensional linear system. In our case, this can be efficiently done by taking advantage of its sparse structure.

Define the N_{app}-th truncation of $\mathbf{K}$ as $\mathbf{K}^{[N_{\mathrm{app}}]} := (\mathbf{K}_{ij}^{[N_{\mathrm{app}}]})_{1\leqslant i,j\leqslant p}$, where $\mathbf{K}_{ij}^{[N_{\mathrm{app}}]} := \boldsymbol{\pi}_{N_{\mathrm{app}}} \cdot \mathbf{K}_{ij} \cdot \boldsymbol{\pi}_{N_{\mathrm{app}}}$. It is represented by the order $p(N_{\mathrm{app}}+1)$ square matrix depicted by blocks in **Figure 5.3a**. By permuting the natural basis $\mathcal{B}_{p,N_{\mathrm{app}}}$ of $(\boldsymbol{\pi}_{N_{\mathrm{app}}} \cdot \mathrm{Y}^1)^p$ into $\mathcal{B}'_{p,N_{\mathrm{app}}}$:

$$\mathcal{B}_{p,N_{\mathrm{app}}} = (T_0e_1, \dots, T_{N_{\mathrm{app}}}e_1, \dots\dots, T_0e_p, \dots, T_{N_{\mathrm{app}}}e_p),$$
$$\mathcal{B}'_{p,N_{\mathrm{app}}} = (T_0e_1, \dots, T_0e_p, \dots\dots, T_{N_{\mathrm{app}}}e_1, \dots, T_{N_{\mathrm{app}}}e_p),$$

$\mathbf{K}^{[N_{\mathrm{app}}]}$ recovers a (ph, pd) almost-banded structure, where $h = \max_{ij} h_{ij}$ and $d = \max_{ij} d_{ij}$ (see **Figure 5.3b**).

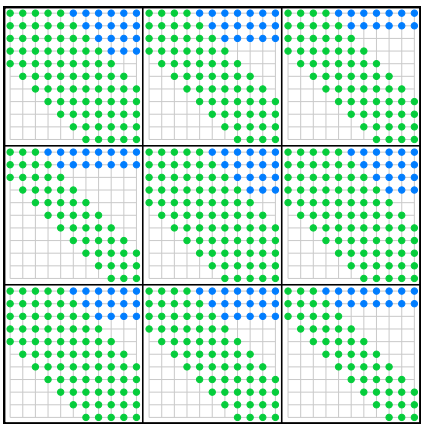
Hence, solving the approximate problem:

$$\Phi + \mathbf{K}^{[N_{\mathrm{app}}]} \cdot \Phi = \Psi$$

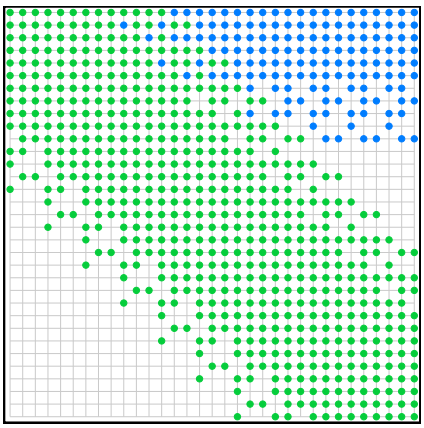
requires $\mathcal{O}(p^3 N_{\mathrm{app}} d^2)$ operations, using the algorithm of [191] for solving almost-banded linear systems (see **Section 4.2**).

5.2.3 ▶ Validation procedure

We extend the validation procedure of **Section 4.3** to the vectorial case. We prove the main **Theorem 5.12** in order to solve **Problem 5.1** in two steps: (1) a Newton-like validation



(a) $\mathbf{K}^{[N_{\text{app}}]}$ in $\mathcal{B}_{p,N_{\text{app}}}$, by blocks ($p = 3$)



(b) $\mathbf{K}^{[N_{\text{app}}]}$ in $\mathcal{B}'_{p,Napp}$ is almost-banded

Figure 5.3: Almost-banded structure of vector-valued integral operators

operator is created and bounded by **Algorithm** IntopVecContract. This first step is independent of the approximation degree N_{app}. (2) The error enclosure of the given approximation is computed by **Algorithm** IntopVecVal, following **Theorems 5.5** and **5.6**.

The goal of this section is to prove the following theorem.

Theorem 5.12 *Algorithms* IntopVecContract *and* IntopVecVal *solve together Problem 5.1 by providing componentwise error enclosures, as tight as desired.*

(5.12 i) *Algorithm* IntopVecContract *only depends on the integral equation (not on the provided approximation). It produces and rigorously bounds a Newton-like validation operator and requires* $\mathcal{O}(p^3 {N_{\text{val}}}^2 d)$ *arithmetic operations.*

(5.12 ii) *Algorithm* IntopVecVal *computes the error enclosures for the approximation and runs in linear time with respect to the maximum degree of the approximations* Φ_i° *and the right-hand sides* Ψ_i*. More precisely, its (arithmetic) complexity is* $\mathcal{O}(p^2 d^2 N_{\text{app}} + pN_{\text{rhs}} + p^2 N_{\text{val}} \min(\max(N_{\text{app}} + d, N_{\text{rhs}}), N_{\text{val}}))$*, where:*

- $N_{\text{app}} := \max_i \deg \Phi_i^\circ$ *and* $N_{\text{rhs}} := \max_i \deg \Psi_i$*;*
- $d := 1 + \max_{ij} \deg k_{ij}(t, s)$*;*
- N_{val} *is the truncation index used to rigorously approximate the problem in finite dimension, as in Section 4.3.*

The previous complexity estimates still involve a truncation index N_{val}, which is directly related to how tight the desired error enclosures have to be. As detailed in **Theorem 5.14**, which extends the discussion led in **Section 4.3.2** to this vectorial setting, complexity estimate of its minimal value ensuring a contracting Newton-like operator is potentially exponential with respect to the magnitude of the coefficients of the integral equation, in the case of stiff LODEs for example. In practice however, this method works efficiently and fully automatically.

Remark 5.13 *The algorithms of Section 5.2 are also implemented in the* ChebValid *C library*[2] *presented in Chapter 3 (the keyword* `_vec` *in the include file names helps to locate*

[2] `https://gforge.inria.fr/projects/tchebyapprox/`

them). The example of Section 5.3 is based on this implementation, as well as applications to space flight dynamics presented in [6] and taken over in Chapter 7.

Newton-like validation operator

Following the quasi-Newton a posteriori validation framework of Section 3.3, Equation (5.10) is transformed into the fixed-point equation:

$$\mathbf{T}\cdot\Phi=\Phi,\qquad \mathbf{T}\cdot\Phi:=\Phi-\mathbf{A}\cdot(\Phi+\mathbf{K}\cdot\Phi-\Psi),\tag{5.11}$$

which is equivalent to (5.10) as soon as $\mathbf{A}:(\mathfrak{l}^1)^p\to(\mathfrak{l}^1)^p$ is injective. Moreover, $\mathbf{T}$ is an affine operator of linear part $\mathcal{D}\mathbf{T}=\mathbf{1}-\mathbf{A}\cdot(\mathbf{1}+\mathbf{K})$. The main challenge is to efficiently compute $\mathbf{A}$ and bound $\|\mathcal{D}\mathbf{T}\|_{(\mathfrak{l}^1)^p}$. This is handled by **Algorithm** IntopVecContract. Similarly to numerical solving, $\mathbf{A}$ approximates $(\mathbf{1}+\mathbf{K}^{[N_{\mathrm{val}}]})^{-1}$, for some truncation order N_{val}. As discussed in the previous chapter, choosing N_{val} is a trade-off between proving $\mathbf{T}$ is contracting (N_{val} must be large enough so that $\|\mathbf{K}-\mathbf{K}^{[N_{\mathrm{val}}]}\|_{(\mathfrak{l}^1)^p}$ is rigorously proved to be sufficiently small) and efficiency requirements.

Once N_{val} is fixed, **Algorithm** IntopVecContract first computes an approximate inverse A in floating-point (lines 1-4). Since $\mathbf{1}+\mathbf{K}^{[N_{\mathrm{val}}]}$ is almost-banded in $\mathcal{B}'_{p,N_{\mathrm{val}}}$, its numerical inverse can be either computed with the numerically stable algorithm of [191], or approximated by a (ph',pd') almost-banded matrix using **Algorithm** AlmostBandedApproxInverse in Section 4.2, in $\mathcal{O}(p^3N_{\mathrm{val}}(h'+d')(h+d))$ operations. The operator $\mathbf{A}$ is defined by extending A to the whole space $(\mathfrak{l}^1)^p$ by the identity.

Second, **Algorithm** IntopVecContract bounds a Lipschitz matrix for $\mathbf{T}$, by $\|\mathcal{D}\mathbf{T}\|_{(\mathfrak{l}^1)^p}:=(\|(\mathcal{D}\mathbf{T})_{ij}\|_{\mathfrak{l}^1})_{1\leqslant i,j\leqslant p}$, block by block, using the triangle inequality:

$$\|\mathcal{D}\mathbf{T}\|_{\mathfrak{l}^1}\leqslant\|\mathbf{1}-\mathbf{A}\cdot(\mathbf{1}+\mathbf{K}^{[N_{\mathrm{val}}]})\|_{\mathfrak{l}^1}+\|\mathbf{A}\cdot(\mathbf{K}-\mathbf{K}^{[N_{\mathrm{val}}]})\|_{\mathfrak{l}^1}.\tag{5.12}$$

The first part of (5.12) is the *approximation error*, measuring how far $\mathbf{A}$ is from the inverse of $\mathbf{1}+\mathbf{K}^{[N_{\mathrm{val}}]}$. This is straightforwardly bounded as Λ^A (lines 5-9) using $\mathcal{O}(p^3N_{\mathrm{val}}(h'+d')(h+d))$ interval arithmetic operations, and the resulting bound takes into account all sources of errors: rounding errors, sparse approximation, etc. Since only additions and multiplications of matrices are involved, the use of interval arithmetic is not critical. However, if needed, the underlying floating-point precision can be increased.

The second part of (5.12) is the *truncation error*, because the truncated operator $\mathbf{K}^{[N_{\mathrm{val}}]}$ only approximates $\mathbf{K}$. Let $\mathbf{E}_{ij}$ be the (i,j) block of $\mathbf{E}:=\mathbf{A}\cdot(\mathbf{K}-\mathbf{K}^{[N_{\mathrm{val}}]})$:

$$\mathbf{E}_{ij}=\sum_{k=1}^{p}\mathbf{A}_{ik}\cdot(\mathbf{K}_{kj}-\mathbf{K}_{kj}^{[N_{\mathrm{val}}]}).\tag{5.13}$$

Algorithm IntopVecContract (lines 10-16) computes $\Lambda^T\geqslant\|\mathbf{E}\|_{(\mathfrak{l}^1)^p}$ by blocks, with the triangle inequality: each subterm of (5.13) is rigorously bounded by **Algorithm** IntopVecTruncError. This algorithm, detailed below, requires $\mathcal{O}((h'+d')(h+d)^2)$ interval arithmetic operations. Hence the computation of Λ^T is in $\mathcal{O}(p^3(h'+d')(h+d)^2)$.

Finally, **Algorithm** IntopVecContract computes $\Lambda=\Lambda^A+\Lambda^T$ and checks that this Lipschitz matrix is convergent to zero, in which case the constructed Newton-like operator $\mathbf{T}$ is contracting. The eigenvalues of Λ can be safely computed with interval arithmetic, for the dimension p is usually small (typically, $p\leqslant 100$).

Proof of Theorem (5.12 i). The detailed description of **Algorithm** IntopVecContract above proves its correctness, and the given complexity estimates for lines 1-4, 5-9 and 10-16 sum to a global complexity of $\mathcal{O}(p^3 N_{\mathrm{val}}(h'+d')(h+d))$ operations. In the worst case, when A is dense ($h'+d' \approx N_{\mathrm{val}}$), we recover the estimate of **Theorem** (5.12 i). □

Truncation error bounding. From Equation (5.13), one needs to bound $\|\mathbf{A}_{ik} \cdot (\mathbf{K}_{kj} - \mathbf{K}_{kj}^{[N_{\mathrm{val}}]})\|_{\mathcal{Y}^1}$, where $\mathbf{A}_{ik}$ is the extension to $\mathcal{Y}^1$ of the order $N_{\mathrm{val}}+1$ matrix A_{ik} by the identity if $i=k$, and zero otherwise. In fact, the case $i=k$ is handled by **Algorithm** IntopTruncError in **Section 4.3**. Therefore, we just need to generalize it to off-diagonal blocks (i.e., $i \neq k$): this is **Algorithm** IntopVecTruncError.

Error enclosures

Finally, **Algorithm** IntopVecVal implements the validation procedure of **Theorems** 5.5 and 5.6 by applying the operator $\mathbf{T}$ to the candidate approximation Φ°, bounding the distance of the resulting polynomial to Φ° and producing componentwise error enclosures to Φ^* with respect to the $\mathcal{Y}^1$ norm.

Proof of Theorem (5.12 ii). **Algorithm** IntopVecVal computes $\Phi^\circ - \mathbf{T}\cdot\Phi^\circ = \mathbf{A}\cdot(\Phi^\circ + \mathbf{K}\cdot\Phi^\circ - \Psi)$. Each P_k (line 1) is a polynomial of degree at most $\max(N_{\mathrm{app}}+d, N_{\mathrm{rhs}})$, and computing its Chebyshev coefficients is in $\mathcal{O}(pd^2 N_{\mathrm{app}} + N_{\mathrm{rhs}})$. Then, the computation of the coefficients of each $A_{ik}\cdot\mathbf{\Pi}_{N_{\mathrm{val}}}\cdot P_k$ (line 3) is in $\mathcal{O}((h'+d')\deg(\mathbf{\Pi}_{N_{\mathrm{val}}}\cdot P_k)) = \mathcal{O}((h'+d')\min(\max(N_{\mathrm{app}}+d, N_{\mathrm{rhs}}), N_{\mathrm{val}}))$.

Finally, the complexity of computing the enclosures (lines 6-7) only depends on p, and is therefore negligible. The overall complexity is:

$$\mathcal{O}(p^2 d^2 N_{\mathrm{app}} + pN_{\mathrm{rhs}} + p^2(h'+d')\min(\max(N_{\mathrm{app}}+d, N_{\mathrm{rhs}}), N_{\mathrm{val}})),$$

which gives the estimate of **Theorem** (5.12 ii) when $h', d' \approx N_{\mathrm{val}}$. □

Estimating N_{val}

The following theorem provides a worst-case estimate for the minimal value of N_{val}. Although theoretically interesting, this exponential bound is over-pessimistic for a wide range of examples.

Theorem 5.14 *Let* $B_{ij} := \sum_{k=0}^{\kappa_{ij}} \|b_{ijk}\|_{\mathcal{Y}^1}$ *and* $B := (B_{ij})_{1\leqslant i,j\leqslant p}$.

(5.14 i) *The following bound estimates the minimum possible value for* N_{val} *making* ***Algorithm*** IntopVecContract *produce a contracting Newton-like operator:*

$$N_{\mathrm{val}} = \mathcal{O}\left(d\rho(B)^2 \exp(2\rho(B))\right),$$

where $\rho(B)$ *denotes the spectral radius of* B.

(5.14 ii) *For a given approximation* Φ° *of* Φ^* *and in order that* ***Algorithm*** IntopVecVal *computes error enclosures* $[\varepsilon_i^-, \varepsilon_i^+]$ *for* $\varepsilon_i = \|\Phi_i^\circ - \Phi_i^*\|_{\mathcal{Y}^1}$ *with* $\varepsilon_i^+/\varepsilon_i^- \leqslant \kappa$ *(for some* $\kappa > 1$*),* N_{val} *for* ***Algorithm*** IntopVecContract *must be at least:*

$$N_{\mathrm{val}} = \mathcal{O}\left(\frac{\nu}{\kappa-1} d\|B\|_\infty^2 \exp(2\|B\|_\infty)\right),$$

Algorithm 5.1 IntopVecContract($\mathbf{K}, N_{\text{val}}, h', d'$) – Create and bound a Newton-like op. $\mathbf{T}$

Input: a polynomial integral operator $\mathbf{K} = (\mathbf{K}_{ij})_{1\leqslant i,j\leqslant p}$ given by the $(b_{ijk})_{0\leqslant k\leqslant \kappa_{ij}}^{1\leqslant i,j\leqslant p}$, a truncation order N_{val}, and optional parameters h', d'.

Output: an approximate inverse $\mathbf{A}$ of $\mathbf{1} + \mathbf{K}^{[N_{\text{val}}]}$ and a certified Lipschitz matrix Λ for $\mathbf{1} - \mathbf{A} \cdot (\mathbf{1} + \mathbf{K})$, or "Fail" if N_{val} not large enough.

▷ *Compute an approximate inverse matrix A.*

$M = (M_{ij})_{1\leqslant i,j\leqslant p} \leftarrow \mathbf{1} + \mathbf{K}^{[N_{\text{val}}]}$, by blocks

$M' \leftarrow M$ in basis $\mathcal{B}'_{p,N_{\text{val}}}$

$A' \leftarrow$ a numerical approximate inverse of M', either dense or (h', d') almost-banded.

$A = (A_{ij})_{1\leqslant i,j\leqslant p} \leftarrow A'$ in basis $\mathcal{B}_{p,N_{\text{val}}}$, by blocks

▷ *Compute the approx error $\Lambda^A = (\lambda_{ij}^A)$ in interval arithmetic*

for $i = 1$ **to** p and $j = 1$ **to** p **do**

  $C \leftarrow \sum_{1\leqslant k\leqslant p} A_{ik}\, M_{kj}$

  if $i = j$ **then** $C \leftarrow C - \mathbf{1}_{N_{\text{val}}+1}$

  $\lambda_{ij}^A \leftarrow \|C\|_1$

end for

▷ *Compute the truncation error $\Lambda^T = (\lambda_{ij}^T)$ in interval arithmetic*

for $i = 1$ **to** p and $j = 1$ **to** p **do**

  $\lambda_{ij}^T \leftarrow 0$

  for $k = 1$ **to** p **do**

    $\delta \leftarrow$ IntopVecTruncError($\mathbf{K}_{jk}, N_{\text{val}}, A_{ik}, i{=}{=}k$)

    $\lambda_{ij}^T \leftarrow \lambda_{ij}^T + \delta$

  end for

end for

▷ *Compute Λ and check if $\mathbf{T}$ contracting.*

$\Lambda \leftarrow \Lambda^A + \Lambda^T$

if $\rho(\Lambda) < 1$ **then**

  return A, Λ

else

  return "Fail"

end if

Algorithm 5.2 IntopVecTruncError($\mathbf{K}, N_{\mathrm{val}}, A, diag$) – Bound the truncation error

Input: a polynomial (one-dimensional) integral operator $\mathbf{K}$ given by the $(b_k)_{0\leqslant k\leqslant\kappa}$, a truncation order N_{val}, a $N_{\mathrm{val}}+1$ order square matrix A, and a Boolean $diag$.

Output: an upper bound δ for $\|\mathbf{A}\cdot(\mathbf{K}-\mathbf{K}^{[N_{\mathrm{val}}]})\|_{\mathrm{L}^1}$, where $\mathbf{A}$ is the extension of A to the whole space L^1 by the identity if $diag = true$, and by zero otherwise.

▷ ***All operations are performed in interval arithmetic***

▷ *Compute* $\delta^{(1)} \geqslant \sup_{\ell\in[\![N_{\mathrm{val}}-d+1,N_{\mathrm{val}}]\!]} B(\ell)$

$\delta^{(1)} \leftarrow 0$
if *diag* **then**
  for $\ell = N_{\mathrm{val}}-d+1$ **to** N_{val} **do**
    $P \leftarrow (\mathbf{1}-\mathbf{\Pi}_{N_{\mathrm{val}}})\cdot\mathbf{K}\cdot T_\ell$
    if $\|P\|_{\mathrm{L}^1} > \delta^{(1)}$ **then** $\delta^{(1)} \leftarrow \|P\|_{\mathrm{L}^1}$
  end for
end if

▷ *Compute* $\delta^{(2)} \geqslant \sup_{\ell\in[\![N_{\mathrm{val}}+1,N_{\mathrm{val}}+d]\!]} B(\ell)$

$\delta^{(2)} \leftarrow 0$
for $\ell = N_{\mathrm{val}}+1$ **to** $N_{\mathrm{val}}+d$ **do**
  $P \leftarrow A\cdot\mathbf{\Pi}_{N_{\mathrm{val}}}\cdot\mathbf{K}\cdot T_\ell$
  if *diag* **then** $P \leftarrow P + (\mathbf{1}-\mathbf{\Pi}_{N_{\mathrm{val}}})\cdot\mathbf{K}\cdot T_\ell$
  if $\|P\|_{\mathrm{L}^1} > \delta^{(2)}$ **then** $\delta^{(2)} \leftarrow \|P\|_{\mathrm{L}^1}$
end for

▷ *Compute* $\delta^{(3)} \geqslant \sup_{\ell\geqslant N_{\mathrm{val}}+d+1} B_D(\ell)$

$\ell_0 \leftarrow N_{\mathrm{val}}+d+1$ **and** $B \leftarrow \sum_{k=0}^{\kappa} \|b_k\|_{\mathrm{L}^1}$
if *diag* **then**
  $P \leftarrow (\mathbf{1}-\mathbf{\Pi}_{N_{\mathrm{val}}})\cdot\mathbf{K}\cdot T_{\ell_0}$
  $\delta^{(3)} \leftarrow \|P\|_{\mathrm{L}^1} + \frac{(\kappa+1)B}{(\ell_0-(\kappa-1))^2}$
else
  $\delta^{(3)} \leftarrow 0$
end if

▷ *Compute* $\delta^{(4)} \geqslant \sup_{\ell\geqslant N_{\mathrm{val}}+d+1} B_I(\ell)$

$B' \leftarrow \sum_{k=0}^{\kappa} \|A\cdot b_k\|_{\mathrm{L}^1}$
$P \leftarrow A\cdot\mathbf{\Pi}_{N_{\mathrm{val}}}\cdot\mathbf{K}\cdot T_{\ell_0}$
$\delta^{(4)} \leftarrow \|P\|_{\mathrm{L}^1} + \frac{(\kappa+1)^3 B'}{(\ell_0^2-(\kappa+1)^2)^2}$

$\delta \leftarrow \max(\delta^{(1)}, \delta^{(2)}, \delta^{(3)}+\delta^{(4)})$
return δ

Algorithm 5.3 IntopVecVal($\mathbf{K}, \Psi, N_{\text{val}}, A, \Lambda, \Phi^\circ$) – Validate a solution of integral equation

Input: a polynomial integral operator $\mathbf{K} = (\mathbf{K}_{ij})_{1\leqslant i,j\leqslant p}$ given by the $(b_{ijk})_{0\leqslant k\leqslant \kappa_{ij}}^{1\leqslant i,j\leqslant p}$, a polynomial right-hand side $\Psi = (\Psi_1, \dots, \Psi_p)$, a truncation order N_{val}, (A, Λ) computed by IntopVecContract (with Λ convergent to 0), and a candidate solution $\Phi^\circ = (\Phi_1^\circ, \dots, \Phi_p^\circ)$.

Output: two vectors of upper and lower bounds ε^+ and ε^- such that $\|\Phi_i^\circ - \Phi_i^*\|_{\mathrm{Y}^1} \in [\varepsilon_i^-, \varepsilon_i^+]$ for $1 \leqslant i \leqslant p$.

▷ ***All operations are performed in interval arithmetic***

1: **for** $k = 1$ **to** p **do** $P_k \leftarrow \Phi_k + \sum_{j=1}^{p} \mathbf{K}_{kj} \cdot \Phi_j^\circ - \Psi_k$
2: **for** $i = 1$ **to** p **do**
3: $\quad Q_i \leftarrow \sum_{k=1}^{p} A_{ik} \cdot \mathbf{\Pi}_{N_{\text{val}}} \cdot P_k + (\mathbf{1} - \mathbf{\Pi}_n) \cdot P_i$
4: $\quad \eta_i \leftarrow \|Q_i\|_{\mathrm{Y}^1}$
5: **end for**
6: $\varepsilon^+ \leftarrow (\mathbf{1} - \Lambda)^{-1}\, \eta$
7: **for** $i = 1$ **to** p **do** $\varepsilon_i^- \leftarrow \left((\mathbf{1} - D_i\, \Lambda)^{-1}\, \eta\right)_i$
8: **return** ε^+ and ε^-

with $\nu := \max_{1\leqslant i\leqslant p} \|\varepsilon\|_\infty/\varepsilon_i$, $\|\varepsilon\|_\infty := \max_{1\leqslant i\leqslant p} \varepsilon_i$ *and* $\|B\|_\infty := \max_{1\leqslant i\leqslant p} \sum_{j=1}^{p} B_{ij}$ *the associated operator norm.*

Proof. (5.14 i) The value of N_{val} must be sufficiently large to ensure that the truncation error $\|(\mathbf{1} + \mathbf{K}^{[N_{\text{val}}]})^{-1} \cdot (\mathbf{K} - \mathbf{K}^{[N_{\text{val}}]})\|_{(\mathrm{Y}^1)^p}$ is a convergent to zero matrix.

- We have as a direct consequence of the one-dimensional case (**Lemma (4.12 ii)**):

$$\|\mathbf{K} - \mathbf{K}^{[N_{\text{val}}]}\|_{(\mathrm{Y}^1)^p} = \mathcal{O}\left(\frac{B}{N_{\text{val}}}\right).$$

- For $i \geqslant 0$, the bound $\|\mathbf{K}^i\|_{(\mathrm{Y}^1)^p} \leqslant (6di + 1)\frac{(2C)^i}{i!}$ is generalized from the one-dimensional case contained in the proof of **Lemma 4.10**, where $C := (C_{ij})_{1\leqslant i,j\leqslant p}$ with $C_{ij} := \sup_{-1\leqslant s,t\leqslant 1} |k_{ij}(t,s)|$ is bounded by B. Since $\mathbf{K}^{[N_{\text{val}}]}$ converges to $\mathbf{K}$, we may approximate:

$$\|(\mathbf{1} + \mathbf{K}^{[N_{\text{val}}]})^{-1}\|_{(\mathrm{Y}^1)^p} \approx \|(\mathbf{1} + \mathbf{K})^{-1}\|_{(\mathrm{Y}^1)^p} = \mathcal{O}\left(dB\exp(2B)\right).$$

- We therefore have:

$$\Lambda = \mathcal{O}\left(\frac{dB^2\exp(2B)}{N_{\text{val}}}\right),$$

where $\Lambda = \|(\mathbf{1} + \mathbf{K}^{[N_{\text{val}}]})^{-1} \cdot (\mathbf{K} - \mathbf{K}^{[N_{\text{val}}]})\|_{(\mathrm{Y}^1)^p}$, and by taking the spectral radius (note that B and $\exp(2B)$ commute):

$$\rho(\Lambda) = \mathcal{O}\left(\frac{d\rho(B)^2\exp(2\rho(B))}{N_{\text{val}}}\right),$$

which gives the estimate for N_{val} to obtain a matrix with spectral radius less than 1.

(5.14 ii) When a maximal overapproximation ratio $\kappa > 1$ is fixed, **Lemma 5.11** provides a condition on the infinite norm $\|\Lambda\|_\infty$ with $\Lambda = \|\mathcal{D}\mathbf{T}\|_{(\mathrm{U}^1)^p}$:

$$\|\Lambda\|_\infty \leqslant \frac{\beta}{2+\beta} = \mathcal{O}\left(\frac{\kappa-1}{\nu}\right), \quad \text{with } \beta := \frac{1}{\nu}\frac{\kappa-1}{\kappa+1}.$$

Again, we focus on the truncation error (since the approximation error can be made as small as desired for a given N_{val}) and we have $\Lambda = \|(\mathbf{1} - \mathbf{K}^{[N_{\mathrm{val}}]})^{-1} \cdot (\mathbf{K} - \mathbf{K}^{[N_{\mathrm{val}}]})\|_{(\mathrm{U}^1)^p}$. Similarly than for the spectral radius, we have:

$$\|\Lambda\|_\infty = \mathcal{O}\left(\frac{d\|B\|_\infty^2 \exp(2\|B\|_\infty)}{N_{\mathrm{val}}}\right).$$

We therefore obtain the desired estimate.

□

5.3 Example and discussion

Consider the following order 1, two-dimensional system, for $x \in [0, a]$ with $a > 0$, whose solutions (depicted in **Figure 5.4**) are highly oscillating functions. Rescale it over $[-1, 1]$ with the change of variable $x = \frac{a}{2}(1+t)$:

$$\begin{cases} y_1' = -x^n y_2 \\ y_2' = x^m y_1 \\ y_1(0) = 1, y_2(0) = 0 \end{cases} \Rightarrow \begin{cases} Y_1' = -\left(\frac{a}{2}\right)^{n+1}(1+t)^n Y_2 \\ Y_2' = \left(\frac{a}{2}\right)^{m+1}(1+t)^m Y_1 \\ Y_1(-1) = 1, Y_2(-1) = 0 \end{cases}. \tag{5.14}$$

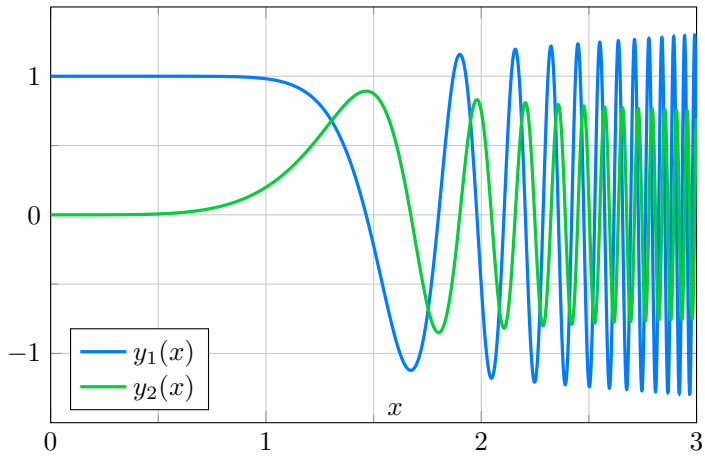


Figure 5.4: Solution of (5.14) with $n = 5$, $m = 4$ and $a = 3$

We give two different integral transforms associated to this equation. The integral transform described in [19] consists in integrating Equation (5.14) once, resulting into an integral

equation for Y with polynomial kernel and right-hand side given by:

$$K(t,s) = \begin{pmatrix} 0 & \left(\frac{a}{2}\right)^{n+1}(1+s)^n \\ -\left(\frac{a}{2}\right)^{m+1}(1+s)^m & 0 \end{pmatrix}, \qquad \Psi(t) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

$K(t,s)$, which is of degree 0 in t, is decomposed over the Chebyshev basis with respect to s into constant polynomials $b_{001}, b_{101}, \dots, b_{n01}$ and $b_{010}, b_{110}, \dots, b_{m10}$.

On the other side, the integral transform of **Proposition 4.5** allows us to validate the derivative $\Phi = Y'$. The polynomial kernel and right-hand side are:

$$K(t,s) = \begin{pmatrix} 0 & \left(\frac{a}{2}\right)^{n+1}(1+t)^n \\ -\left(\frac{a}{2}\right)^{m+1}(1+t)^m & 0 \end{pmatrix}, \qquad \Psi(t) = \begin{pmatrix} \left(\frac{a}{2}\right)^{n+1}(1+t)^n \\ 0 \end{pmatrix}.$$

Now, $K(t,s)$ is of degree 0 with respect to s, giving two polynomials b_{001} and b_{010} of respective degrees n and m.

Let's now focus on the first integral transform, with $n = 5$, $m = 4$, $a = 3$. Using the spectral method explained in **Section 5.2.2** and implemented in ChebValid, we fix an approximation degree $N_{\text{app}} = 100$ and obtain numerical approximations Y_1° and Y_2°, that must now be validated. The whole implemented procedure automatically computes and bounds for increasing values of N_{val} the Newton-like operator $\mathbf{T}$ associated to the truncated operator $\mathbf{K}^{[N_{\text{val}}]}$. The approximate inverse is computed as an $(2h', 2d')$ almost-banded order $2(N_{\text{val}}+1)$ matrix. This process stops as soon as the total Lipschitz matrix returned by **Algorithm** IntopVecContract has a spectral radius less than 1. In case of failure of IntopVecContract, the procedure is relaunched with $N_{\text{val}} \leftarrow 2N_{\text{val}}$. For this example, we obtain $N_{\text{val}} = 1664$, $h' = 48$ and $d' = 304$, giving the following Lipschitz matrix:

$$\Lambda = \begin{pmatrix} 9.73 \cdot 10^{-4} & 9.89 \cdot 10^{-2} \\ 3.60 \cdot 10^{-2} & 9.92 \cdot 10^{-2} \end{pmatrix}, \qquad \rho(\Lambda) = 6.06 \cdot 10^{-2}.$$

The last step is performed by **Algorithm** IntopVecVal. Given the numerical approximations Y_1° and Y_2°, it computes $\eta = \|Y^\circ - \mathbf{T} \cdot Y^\circ\|_{(\ell^1)^2}$ (the example gives $\eta_1 = 3.20 \cdot 10^{-3}$ and $\eta_2 = 1.91 \cdot 10^{-3}$) and outputs the error enclosures given by **Theorems 5.5** and **5.6**:

$$\varepsilon_1^- = 2.99 \cdot 10^{-3}, \qquad \varepsilon_1^+ = 3.41 \cdot 10^{-3},$$
$$\varepsilon_2^- = 1.78 \cdot 10^{-3}, \qquad \varepsilon_2^+ = 2.04 \cdot 10^{-3}.$$

This whole process for this example takes about 30 seconds on a modern computer.

Comparison with decoupling/desingularization

In the case of polynomial coefficients, an alternative consists in decoupling the system to obtain p scalar LODEs of order p, at the cost of introducing singularities in the equations. As an example, the first component y_1 in (5.14) satisfies the following differential equation:

$$xy_1'' - ny_1' + x^{n+m+1}y_1 = 0. \tag{5.15}$$

This equation is *singular* (its leading coefficient vanishes at 0), so our validation method cannot be used. However, with *desingularization* techniques [1], one obtains a higher order but non-singular equation, whose set of solutions (strictly) contains the ones of the singular equation.

In our example, by differentiating Equation (5.15) n times and dividing the result by x:

$$y_1^{(n+2)} + \frac{1}{x}\frac{\mathrm{d}^n}{\mathrm{d}x^n}(x^{n+m+1}y_1) = 0. \tag{5.16}$$

By inverting the roles of n and m, one obtains a similar equation for y_2. Hence, validating the approximation y of (5.14) can be done with the validation algorithm IntopVal for the scalar case. Several caveats must therefore be raised. Applying the integral operator of **Proposition 4.5** results into a totally intractable problem, since the minimal value for proving that $\mathbf{T}$ is contracting is far too large (in practice, we stopped at $N_{\mathrm{val}} \simeq 10^6$). This is due to the fact that this transform is used to validate the last derivative $y_1^{(n+2)}$, which increases very rapidly due to the highly oscillating behavior of y_1. On the other hand, the integral transform of [19] yields a far more tractable problem: a truncation order $N_{\mathrm{val}} = 750$ is sufficient for our example. However, Equation (5.16) is very ill-conditioned because of the factorial terms created by the n differentiations. For instance, with classical double precision (53 bits), the scalar validation procedure is able to produce and bound a contracting Newton-like operator $\mathbf{T}$ (**Algorithm** IntopContract), but **Algorithm** IntopVal outputs an upper bound $\varepsilon_1^+ = 2.57$, which is 3 orders of magnitude larger than what was found with the vector-valued validation **Algorithm** IntopVecVal.

The non D-finite case. In the case of nonpolynomial coefficients, there is no general method to decouple and desingularize the system. Moreover, these coefficients may not be known exactly, but only given as polynomial approximations together with rigorous error bounds. We believe that in such a general case, the vector-valued approach presented in this article is essential to approximate and validate the solution.

An example of a successful application of this validation to a "real life" linear system of LODEs with non-polynomial coefficients is given in **Chapter 7**.

5.4 Conclusion and future directions

In **Chapters 4** and **5**, we proposed a generic efficient algorithm for computing rigorous polynomial approximations for LODEs (or coupled systems of LODEs). We focused on both its theoretical and practical complexity analysis. For this, firstly, we studied theoretical properties like compactness, convergence, invertibility of associated linear integral operators and their truncations over $\mathrm{Ч}^1$, the coefficient space of Chebyshev series. Then, we focused on the almost-banded matrix structure of these operators, which allowed for very efficient numerical algorithms for both the numerical solutions of LODEs and the rigorous computation of the approximation error. More specifically, the proposed a posteriori validation algorithm is based on a quasi-Newton method, which benefits from the almost-banded structure of intervening operators. In the vector-valued case, the extension of Banach fixed-point theorem in **Section 5.1** moreover yields componentwise tight error bounds, instead of a global one. Finally, several representative examples showed the advantages of our algorithms as well as their theoretical

and practical limits. Computations were carried out using the ChebValid C library for RPAs[3] presented in **Chapter 3**.

Several extensions of this work really deserve being investigated:

- A work in progress is to rewrite the Picard iterations based validation method presented in **[19]** as a quasi-Newton validation technique. Then, using our current almost-banded operator based algorithms, we will be able to generalize the method in **[19]** to non-homogeneous and non-polynomial LODEs with a better complexity bound, relying on a more involved analysis of the iterated kernels.

- We also consider the generalization to other families of orthogonal polynomials, such as Legendre polynomials, or Hermite and Laguerre polynomials over unbounded intervals. In fact, orthogonal polynomials always satisfy a three-term-recurrence (see **Section 2.2.3**), so that the multiplication and integration formulas remain similar, which should produce similar almost-banded integral operators. However, the operator theoretical aspects (compactness, convergence of truncations, etc.) are more challenging for unbounded intervals. Further investigations are needed.

- The propagation of uncertain initial conditions via LODEs may also be explored based on our current techniques.

- Another challenging direction is non-linear ODEs and (linear) PDEs. In both cases however, we have to rely on a multivariate approximation theory with orthogonal polynomials (such theories exist but are not unique and depend on the domain of approximation) and the theory for such differential equations are far less structured than the easy linear univariate case. In particular, the time complexity of such extensions may be huge compared to the present case.

- On the formal proof side, we intend to formalize these algorithms in the Coq development[4] presented in **Chapter 3**, to guarantee both the theoretical correctness of that method and the soundness of its current C implementation.

[3] `https://gforge.inria.fr/projects/tchebyapprox/`
[4] `http://perso.ens-lyon.fr/florent.brehard/chebapprox/`

Part III

An Eclectic Mix of Related Problems and Applications

A New Lower Bound on the Hilbert Number for Quartic Systems

6

Si la logique est l'hygiène du mathématicien, ce n'est pas elle qui lui fournit sa nourriture ; le pain quotidien dont il vit, ce sont les grands problèmes.
— André Weil

This chapter takes over the main lines of an ongoing work with Nicolas Brisebarre, Mioara Joldes and Warwick Tucker. Therefore, in this chapter, *we* refers to the four of us. It consists in an interesting application of symbolic-numeric methods introduced in the previous chapters to a computer-assisted proof for the existence of limit cycles in the setting of Hilbert's 16th problem. More specifically, we prove that there exists a planar, polynomial vector field of degree four exhibiting (at least) 24 limit cycles, which improves the previously best lower bound of 22.

6.1 Introduction and global setting

In 1900, at the International Congress of Mathematics held in Paris, David Hilbert presented ten open problems in mathematics, and later published a more comprehensive list of 23 problems [107] aimed at challenging the mathematical community. Today, most of the Hilbert problems have been resolved (two of them were deemed to be unresolvable), but a few ones still remain open: one of these is Hilbert's 16th problem.

Hilbert's 16th problem has two distinct parts: one in real algebraic geometry, and one in dynamical systems. We will address the latter which asks for $\mathcal{H}(n)$ – the maximal number of limit cycles (i.e., isolated periodic orbits) the family of two-dimensional polynomial vector fields of (total) degree at most n can display:

Consider the differential equation in $\mathbb{R}^2$*:*

$$\begin{cases} \dot{x} = P_n(x, y), \\ \dot{y} = Q_n(x, y), \end{cases} \tag{6.1}$$

where P_n and Q_n are polynomials of degree at most n. Is there a minimal upper bound $\mathcal{H}(n)$ on the number of limit cycles the system (6.1) can have, that only depends on the degree n?

Note that the bound $\mathcal{H}(n)$ should be uniform, that is, it should not depend on the particular polynomial vector field, only on its degree n. As of today, this question is not resolved even in the simplest case $n = 2$. Even finding realistic *lower* bounds for $\mathcal{H}(n)$ appears to be very hard.

The main result of this paper is the following lower bound on $\mathcal{H}(n)$ in the quartic setting.

Theorem 6.1 *There exist bivariate, quartic polynomials H, P and Q such that, for sufficiently small values of $\varepsilon > 0$, the system*

$$\begin{cases} \dot{x} = -yH_y(x,y) + \varepsilon P(x,y), \\ \dot{y} = yH_x(x,y) + \varepsilon Q(x,y), \end{cases} \tag{6.2}$$

exhibits 24 limit cycles. Hence $\mathcal{H}(4) \geqslant 24$.

Here we are using the notation $H_x = \frac{\partial H}{\partial x}$ and $H_y = \frac{\partial H}{\partial y}$ for brevity. The three polynomials H, P, and Q appearing in the theorem are explicit, see **Section 6.2** for details. Note that the previously known best lower bound was $\mathcal{H}(4) \geqslant 22$ [56].

Moreover, the ongoing work presented in **Section 6.4** aims at proving that 24 is a local upper bound on the number of limit cycles. For now, we can only state the following conjecture.

Conjecture 6.2 *For the same polynomial H as used in **Theorem 6.1**, there is no quartic perturbation such that the differential system (6.2) gives rise to more than 24 limit cycles.*

This conjecture being proved would imply that we have obtained the maximal number of limit cycles that can arise when perturbing the integrable system under study. Of course, there might very well exist other quartic polynomials H that produce more limit cycles under perturbations of the same degree.

Before describing the details of our set of tools and proof techniques used for establishing **Theorem 6.1**, we take a step back and provide a short historical perspective of Hilbert's 16th problem together with a summary of known results.

6.1.1 ▶ Historical remarks

Hilbert's 16th problem has a remarkable history; when it comes to finding upper bounds for $\mathcal{H}(n)$, very little progress has been made since Hilbert's seminal talk in 1900.

Some historical landmarks for Hilbert's problem

1923 Dulac [74] published a very important piece of work, stating that a *single* polynomial vector field has only finitely many limit cycles.

1955 Petrovskii and Landis [196] stated that $\mathcal{H}(2) = 3$ and $\mathcal{H}(n)$ grows cubically in n.

1962 The claims by Petrovskii and Landis were disproved by Novikov and Ilyashenko [146].

1981 A serious gap was found in Dulac's proof [117].

1991 Dulac's result was given new (extremely complicated) proofs by Ilyashenko [118] and Écalle [77].

In a survey article [116], Ilyashenko commented: *Thus, after eighty years of development, our knowledge on Hilbert's 16th problem was almost the same as at the time when the problem was stated.*

When it comes to establishing *lower* bounds for $\mathcal{H}(n)$, there has been some progress:

1979 Both S. Shi [229], and L. Chen and M. Wang [52] found examples for $\mathcal{H}(2) \geqslant 4$.

1987 L. Li and Q. Huang [155] constructed a cubic system with 11 limit cycles: $\mathcal{H}(3) \geqslant 11$.

1988 R. Roussarie [216] reduced the question of *uniform* finiteness in the quadratic case to proving finite cyclicity of 121 *graphics*. Today ca 85 of these have been successfully dealt with.

2003 J. Li [156] proved that $\mathcal{H}(n) \geqslant \frac{1}{4}(n+1)^2(1.442695 \ln(n+1) - \frac{1}{6}) + n - \frac{2}{3}$.

2005 C. Christopher [56] gave an example for $\mathcal{H}(4) \geqslant 22$. (This bound was believed to have been improved in [126] which later turned out to be erroneous)

2009 C. Li, C. Liu and J. Yang gave an example for $\mathcal{H}(3) \geqslant 13$.

It is worth mentioning that the proofs of these results are very technical and long. Furthermore, it is not known whether any of the obtained lower bounds are sharp.

In light of the lack of progress regarding bounds for $\mathcal{H}(n)$, in the mid-seventies, V.I. Arnold [10, 11] proposed to study a restricted version of the original problem, now known as the *infinitesimal* (or *weak*, or *tangential*) Hilbert's 16th problem. Rather than considering the class of all polynomial vector fields of a certain degree, Arnold suggested that only small perturbations of Hamiltonian polynomial vector fields be considered. Thus, the corresponding question can be asked:

Consider the differential equation in $\mathbb{R}^2$*:*

$$\begin{cases} \dot{x} = -H_y(x,y) + \varepsilon P_n(x,y), \\ \dot{y} = H_x(x,y) + \varepsilon Q_n(x,y), \end{cases} \tag{6.3}$$

where $H(x,y)$ *is a polynomial of degree at most* m*,* P_n *and* Q_n *are polynomials of degree at most* n*, and* $\varepsilon \neq 0$ *is small. Is there a bound* $\mathcal{Z}(m,n)$ *on the number of limit cycles the system* (6.3) *can have (for small* ε*), that only depends on the degrees* m *and* n*?*

Note that we immediately have the lower bound $\mathcal{Z}(n) \stackrel{\text{def}}{=} \mathcal{Z}(n+1,n) \leqslant \mathcal{H}(n)$.

For the infinitesimal problem, significant progress has been made, and we have a partial understanding of how polynomial Hamiltonian systems behave under small perturbations.

Some historical landmarks for the infinitesimal problem

1984 Varchenko [254] and Khovanskii [138] (independently) proved that $\mathcal{Z}(n,m) < \infty$.

2006 A uniform proof for $\mathcal{Z}(2) = 2$ appeared in 2006 by Chen, Li, Llibre, and Zhang [51]. This result was based on special cases established during the period 1994–2002.

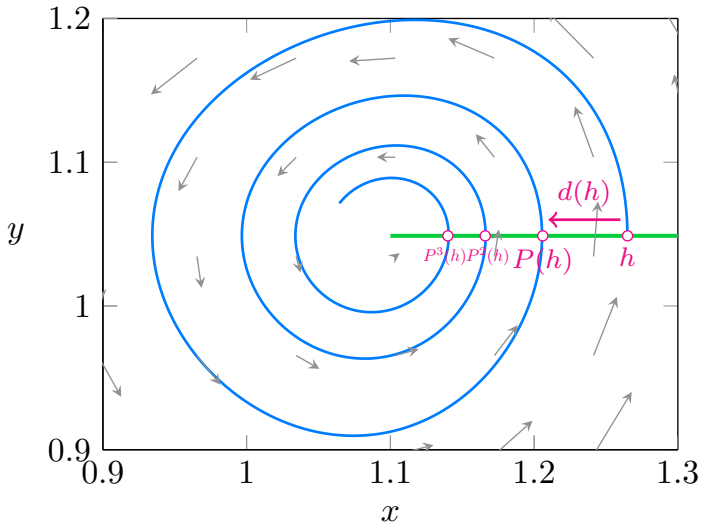
2010 Binyamini, Novikov, and Yakovenko [26] proved an upper bound: $\mathcal{Z}(n) \leqslant 2^{2^{p(n)}}$, where p is an "explicit" polynomial of degree no greater than 61.

Some other results are: $\mathcal{Z}(3) \geqslant 13$ [154], $\mathcal{Z}(5) \geqslant 27$ [128], $\mathcal{Z}(7) \geqslant 53$ [127], and $\mathcal{Z}(11) \geqslant 121$ [258].

6.1.2 ▶ Abelian integrals

The study of perturbed Hamiltonian systems depends heavily on a theorem by Poincaré and Pontryagin, that makes a strong connection between the existence of a limit cycle and a zero of a certain *Abelian integral*.

The *Poincaré return map* is the key tool to understand this connection. Consider the unperturbed Hamiltonian system and take a transversal Σ, that is a portion of a curve crossing non-tangentially a foliation of periodic orbits of this unperturbed system. This transversal may be parameterized by the parameter h (the energy level) of the unperturbed system for some domain $h_1 \leqslant h \leqslant h_2$. By continuity, for a sufficiently small ε, every trajectory originating from a point $\Sigma(h)$ of the transversal in the perturbed Hamiltonian system will cross Σ again. The Poincaré return map Π associates to h and ε the parameter $\Pi(h, \varepsilon)$ corresponding to the point of return to Σ, and $d(h, \varepsilon) = \Pi(h, \varepsilon) - h$ is called the *displacement function*. Clearly, the point $\Sigma(h)$ belongs to a periodic orbit of the perturbed system if and only if $d(h, \varepsilon) = 0$, and this is a limit cycle if and only if it is isolated.



■ **Figure 6.1**: Poincaré return map Π (=: P) and displacement function d.

Zeros of Abelian integrals are related to limit cycles in the following way: given a Hamiltonian system and a perturbation (6.3) the Abelian integral over the *oval* $\Gamma(h)$ is defined as

$$\mathfrak{I}(h) = \int_{\Gamma(h)} P_n(x, y)\, dy - Q_n(x, y)\, dx.$$

Here $\Gamma(h)$ is a closed curve (perhaps one of several) making up the level set $H^{-1}(h)$ of the

Hamiltonian. The Poincaré-Pontryagin theorem roughly states that $d(h,\varepsilon) \approx \varepsilon\mathfrak{I}(h)$ for small values of ε (see **Theorem 6.3** for more details). Hence, the number of isolated zeros of $\mathfrak{I}(h)$ where a change of sign occurs (in particular, simple zeros of $\mathfrak{I}(h)$) provides a lower bound for the number of limit cycles of (6.3) that exist for small $\varepsilon > 0$. Considering zeros of $\mathfrak{I}(h)$ of higher multiplicity it is possible to get an *upper* bound on the number of limit cycles that can bifurcate from the unperturbed periodic orbit(s) $\Gamma(h)$, see [57].

The function $\mathfrak{I}(h)$ can be decomposed into a linear combination

$$\mathfrak{I}(h) = \alpha_0\mathfrak{I}_0(h) + \alpha_1\mathfrak{I}_1(h) + \cdots + \alpha_{m-1}\mathfrak{I}_{m-1}(h), \tag{6.4}$$

where each α_k depends on the coefficients of P_n and Q_n, which are considered as parameters, and each $\mathfrak{I}_k$ is an Abelian integral with the 1-form $\omega = x^i y^j dx$ or $\omega = x^i y^j dy$. Therefore the problem of bounding the number of limit cycles of (6.3) is equivalent to bounding the number of zeros of any function in the linear span of $\mathfrak{I}_0$, $\mathfrak{I}_1 \ldots, \mathfrak{I}_{m-1}$.

6.1.3 ▶ Our approach

In [126], T. Johnson constructs a quartic pseudo-Hamiltonian vector field together with a perturbation (defined by appropriate coefficients α_i as in Equation (6.4)). Using a rigorous validation integration routine, he claims to prove the existence of 26 limit cycles, thus surpassing the previously known record $\mathcal{H}(4) \geqslant 22$ [57]. Unfortunately, a bug in his implementation led him to observe more zeros in the Abelian integral than what actually exists.

In the present work, we take over the quartic vector field of [126], which is introduced in details in **Section 6.2**, and give new values for the coefficients of the perturbation to recover as many limit cycles as possible. We found 24 such ones, which, although less than 26, is larger than the record 22. We stress out the fact that the evaluation of Abelian integrals, which guarantees the existence of 24 limit cycles, is not only carried out using our ChebValid C library[1], but also certified by the Coq development[2] [40] presented in **Chapter 3**. The goal is to provide a proof of **Section 6.1** with the highest confidence level. The whole approach, from the computation of the coefficients of the perturbation to the rigorous *and* certified evaluation of the Abelian integrals, is summarized in **Section 6.3**.

Finally, we address in **Section 6.4** the following natural question: is it possible to find even more limit cycles for the same vector field, by finding other values for the coefficients of the perturbation? For this end, we will need the notion of *Wronskian* to investigate the maximal possible number of zeros of linear combinations of Abelian integrals. Consequently, *continuous* rigorous representations of the Abelian integrals are necessary.

[1] `https://gforge.inria.fr/projects/tchebyapprox/`
[2] `http://perso.ens-lyon.fr/florent.brehard/chebapprox/`

6.2 | Construction of a perturbed system

In order to provide a lower bound for $\mathcal{H}(4)$, we investigate the limit cycles of a degree 4 perturbed *pseudo*-Hamiltonian system, presented in **Section 6.2.2** and based on the potential function H defined in **Section 6.2.1**. Since this system is not Hamiltonian, this does not result into a lower bound for $\mathcal{Z}(4)$. However, this system is still *integrable*, and the *generalized* Poincaré-Pontryagin theorem (**Theorem 6.3**) allows us to use similar techniques based on Abelian integrals to determine a perturbation maximizing the number of limit cycles. In **Section 6.2.3**, we give the explicit parameterizations of the resulting ovals, which will be used in the rigorous evaluation of the Abelian integrals in **Section 6.3**.

6.2.1 ▶ The potential function H

The potential function used throughout this article is defined as in **[126]**, up to a constant:

$$H(x,y) = (x^2 - X_0)^2 + (y^2 - Y_0)^2, \tag{6.5}$$

where X_0 and Y_0 are constants satisfying $0 < X_0 < Y_0$ (the choice $X_0 < Y_0$ will be explained below).

The level set associated to the parameter $h \geqslant 0$, represented in the (x^2, y^2) plane, is the portion of the circle of center (X_0, Y_0) and radius $r = \sqrt{h}$ located in the positive quadrant (see **Figure 6.2a**). In the (x, y) plane, this results into the ovals depicted in **Figure 6.2b**. In particular, it is symmetric with respect to both the x and y axes. More specifically, three cases for $h > 0$ are to be distinguished:

- When $h \in (0, X_0^2)$ (i.e., $r \in (0, X_0)$), the circle defined by (x^2, y^2) entirely lies in the positive quadrant. In the (x, y) plane, this results into four symmetric ovals that we call *small ovals*:

$$\begin{aligned} \Gamma^{++}(h) &= H^{-1}(h) \cap \mathbb{R}_{\geqslant 0} \times \mathbb{R}_{\geqslant 0}, & \Gamma^{+-}(h) &= H^{-1}(h) \cap \mathbb{R}_{\geqslant 0} \times \mathbb{R}_{\leqslant 0}, \\ \Gamma^{-+}(h) &= H^{-1}(h) \cap \mathbb{R}_{\leqslant 0} \times \mathbb{R}_{\geqslant 0}, & \Gamma^{--}(h) &= H^{-1}(h) \cap \mathbb{R}_{\leqslant 0} \times \mathbb{R}_{\leqslant 0}. \end{aligned} \tag{6.6}$$

- When $h \in (X_0^2, Y_0^2)$ (i.e., $r \in (X_0, Y_0)$), a portion of this circle crosses the y-axis. The four small ovals merge into two symmetric *big ovals*:

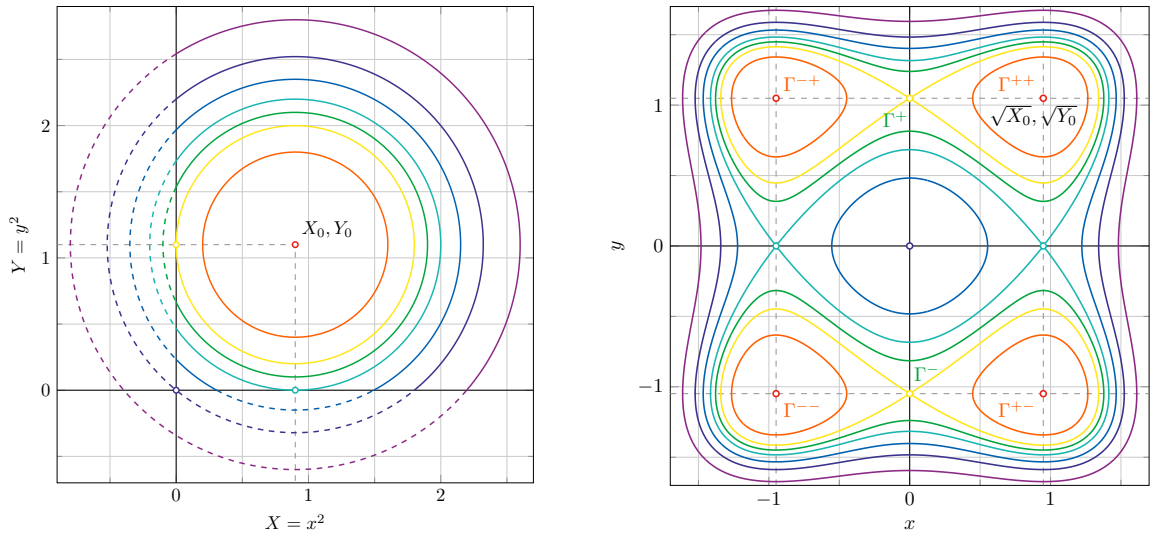
$$\Gamma^{+}(h) = H^{-1}(h) \cap \mathbb{R} \times \mathbb{R}_{\geqslant 0}, \qquad \Gamma^{-}(h) = H^{-1}(h) \cap \mathbb{R} \times \mathbb{R}_{\leqslant 0}. \tag{6.7}$$

- For $h > Y_0^2$ (i.e., $r > Y_0$), the resulting ovals (one external and one internal for $Y_0 < r < \sqrt{X_0^2 + Y_0^2}$, and only one external for $r > \sqrt{X_0^2 + Y_0^2}$) all cross the x-axis. Because of the rescaling by y used in the pseudo-Hamiltonian system given in next section, these ovals do not correspond to periodic orbits but to *heteroclinic* orbits.

Notice moreover that for the limit case $r = X_0$ (resp. Y_0), the small (resp. big) ovals meet at a singular point, which will be an equilibrium point in the pseudo-Hamiltonian system, resulting

into a *homoclinic* (resp. *heteroclinic*) orbit. In the rest of the article, we will therefore focus on the first two cases, corresponding to the four small ovals and the two big ovals.

Note that if we assume $Y_0 \leqslant X_0$, we would only have one useful case (the four small ovals), instead of two, to exploit.



(a) Circles in the (x^2, y^2) plane

(b) Ovals in the (x, y) plane

■ **Figure 6.2**: Level curves of the potential function H

6.2.2 ▶ A pseudo-Hamiltonian system

We define from the potential function (6.5) the following perturbed pseudo-Hamiltonian system:

$$\begin{cases} \dot{x} = -y\partial_y H(x,y) + \varepsilon g_1(x,y) = -4y^2(y^2 - Y_0) + \varepsilon g_1(x,y), \\ \dot{y} = \quad y\partial_x H(x,y) + \varepsilon g_2(x,y) = 4xy(x^2 - X_0) + \varepsilon g_2(x,y), \end{cases} \tag{6.8}$$

where g_1 and g_2 are polynomials of degree 4. The unperturbed system is obtained from the usual Hamiltonian system by a rescaling by the second variable y, whence the denomination *pseudo*-Hamiltonian. The trajectories still follow the level curves of H, but now the vector field vanishes over the whole horizontal line $y = 0$ (all the points on this line are equilibrium points).

At first sight, one might get the impression that this rescaling by y just reduces the number of periodic orbits, since a periodic orbit crossing the line $y = 0$ is transformed into two heteroclinic orbits. However, when regarding the perturbed system, the following adaptation of the Poincaré-Pontryagin Theorem (see [57] for a proof) to this pseudo-Hamiltonian setting offers more possibilities than the Hamiltonian case:

Theorem 6.3 (Generalized Poincaré-Pontryagin Theorem) *Let $H : \mathbb{R}^2 \to \mathbb{R}$ be a real analytic potential function, $g_1, g_2 : \mathbb{R}^2 \to \mathbb{R}$ real analytic functions, $\varepsilon > 0$, and $\sigma : \mathbb{R}^2 \to \mathbb{R}$ a* rescaling factor. *Consider the perturbed pseudo-Hamiltonian system:*

$$\begin{cases} \dot{x} = -\sigma(x,y)\partial_y H(x,y) + \varepsilon g_1(x,y), \\ \dot{y} = \quad \sigma(x,y)\partial_x H(x,y) + \varepsilon g_2(x,y). \end{cases}$$

For an oval of H of level h over which σ does not vanish, define the Abelian integral

$$\mathfrak{I}(h) = \int_{\Gamma(h)} \frac{g_1(x,y)\mathrm{d}y - g_2(x,y)\mathrm{d}x}{\sigma(x,y)}.$$

*Then the displacement function $d(h,\varepsilon)$ (as described in **Section 6.1.2**) is approximated as*

$$d(h,\varepsilon) = \varepsilon\mathfrak{I}(h) + O(\varepsilon^2), \qquad \text{as } \varepsilon \to 0.$$

In particular, when $\mathfrak{I}(h) \neq 0$, $d(h,\varepsilon)$ and $\mathfrak{I}(h)$ have the same (strict) sign for small $\varepsilon > 0$.

In our case, this allows us to have integrands of the form $x^i y^j/y\,\mathrm{d}x$ and $x^i y^j/y\,\mathrm{d}y$ with $i + j \leqslant 4$, which are *not* polynomials for $j = 0$. We therefore have more flexibility than the pure Hamiltonian setting. To each such perturbation $g = (g_1, g_2)$, we can associate Abelian integrals on small and big ovals, where the line integral is taken in the usual trigonometric orientation:

$$\mathfrak{I}^{\square\diamond}(h) = \int_{\Gamma^{\square\diamond}(h)} \frac{g_1(x,y)\mathrm{d}y - g_2(x,y)\mathrm{d}x}{y}, \qquad (\square,\diamond) \in \{+,-\}^2, \qquad h \in (0, X_0^2),$$
$$\mathfrak{I}^{\diamond}(h) = \int_{\Gamma^{\diamond}(h)} \frac{g_1(x,y)\mathrm{d}y - g_2(x,y)\mathrm{d}x}{y}, \qquad \diamond \in \{+,-\}, \qquad h \in (X_0^2, Y_0^2).$$

As in [126], we only consider the following restricted perturbation:

$$g_1(x,y) = 0, \qquad g_2(x,y) = \alpha_{00} + \alpha_{20}x^2 + \alpha_{22}x^2y^2 + \alpha_{40}x^4 + \alpha_{04}y^4, \quad \alpha_{ij} \in \mathbb{R}.$$

This choice is guided by the following requirements:

- *Symmetry*: We decide to keep only symmetric perturbations, meaning that the four Abelian integrals $\mathfrak{I}^{\square\diamond}(h)$ must be equal, up to the sign. Moreover, $\mathfrak{I}^{+\diamond}(h)$ and $\mathfrak{I}^{-\diamond}(h)$ must have the same sign, otherwise $\mathfrak{I}^{\diamond}(h) = 0$ for $h \in (X_0^2, Y_0^2)$ on big ovals:

$$\mathfrak{I}^{++}(h) \;=\; \mathfrak{I}^{-+}(h) \;=\; (-1)^e\mathfrak{I}^{--}(h) \;=\; (-1)^e\mathfrak{I}^{+-}(h), \qquad e \in \{0,1\}. \tag{6.9}$$

  With these conditions, one just needs to investigate the sign alternations of $\mathfrak{I}^{++}(h)$ on the small ovals $\Gamma^{++}(h)$ for $h \in (0, X_0^2)$, and $\mathfrak{I}^{+}(h)$ on the big ovals $\Gamma^{+}(h)$ for $h \in (X_0^2, Y_0^2)$. We simply write $\mathfrak{I}(h)$ and $\Gamma(h)$ in that case.

- *Linear independence*: The Abelian integrals corresponding to the monomials appearing in g_1 and g_2 must form an independent family.

Proposition 6.4 *The five Abelian integrals corresponding to the five monomials of (6.2.2) are linearly independent and satisfy the symmetry requirements (6.9). Moreover, this family is maximal in the sense that adding another monomial in g_1 or g_2 gives rise to an extra Abelian integral that violates the symmetry requirements or the linear independence.*

Proof. Consider $g = (g_1, g_2)$ a generic polynomial perturbation of total degree at most 4.

- The symmetry requirements (6.9), are satisfied whenever:

$$\begin{aligned} x \mapsto -x \quad &: \quad \frac{g_1(x,y)\mathrm{d}y - g_2(x,y)\mathrm{d}x}{y} = \frac{-g_1(-x,y)\mathrm{d}y - g_2(-x,y)\mathrm{d}x}{y}, \\ y \mapsto -y \quad &: \quad \frac{g_1(x,y)\mathrm{d}y - g_2(x,y)\mathrm{d}x}{y} = (-1)^e \frac{-g_1(x,-y)\mathrm{d}y - g_2(x,-y)\mathrm{d}x}{y}. \end{aligned}$$

  This means that g_1 (respectively g_2) only contain monomials of the form $x^{i_1}y^{j_1}$ (respectively $x^{i_2}y^{j_2}$), with $i_1 \equiv 1 \mod 2$, $i_2 \equiv 0 \mod 2$, $j_1 = 1 - e \mod 2$ and $j_2 = e \mod 2$. The perturbation (6.2.2) clearly satisfies these requirements, with the choice $e = 0$. Moreover, adding a new monomial in g_1 or g_2 is inconsistent with the symmetry requirements if the parities of its exponents do not follow the same rule (with $e = 0$).

- A straightforward application of Green's theorem shows that the Abelian integral associated to $(g_1/y)\mathrm{d}y$ and $(g_2/y)\mathrm{d}x$ are linearly dependent if $\partial_x \cdot (g_1/y)$ and $\partial_y \cdot (g_2/y)$ are equal up to a multiplicative constant. But if i_1 and j_1 are odd with $i_1 + j_1 \leqslant 4$, then we can easily check that there exists i_2 and j_2 even with $i_2 + j_2 \leqslant 4$ such that $g_1 = x^{i_1}y^{j_1}/y$ and $g_2 = x^{i_2}y^{j_2}/y$ satisfy the above condition. Hence, we can without loss of generality set $g_1 = 0$, and only consider a perturbation g_2 along the y-axis.

- Following the previous remarks, the only candidate monomial we could add to the five ones 1, x^2, x^2y^2, x^4, y^4 of g_2 is y^2. However, the linear independence is violated:

$$\int_{\Gamma(h)} \frac{3x^4 - 3X_0x^2 - y^4 + 3Y_0y^2}{y}\mathrm{d}x = 0, \qquad h \in (X_0^2, Y_0^2).$$

  Indeed, Green's theorem allows us to rewrite the left-hand side as:

$$\begin{aligned} &\int_{\mathrm{Int}(\Gamma(h))} \frac{3}{y^2}\left(x^2(x^2 - X_0) + y^2(y^2 - Y_0)\right)\mathrm{d}x\mathrm{d}y \\ &\quad = \int_{\mathcal{D}((X_0,Y_0),r)} \frac{3}{4Y\sqrt{XY}}\left(X(X - X_0) + Y(Y - Y_0)\right)\mathrm{d}X\mathrm{d}Y, \qquad \text{with } r = \sqrt{h} \\ &\quad = \frac{3}{2}\int_{\mathcal{D}((X_0,Y_0),r)} \left(-\partial_Y G\,(X - X_0) + \partial_X G\,(Y - Y_0)\right)\mathrm{d}X\mathrm{d}Y, \quad \text{with } G = \sqrt{X/Y} \\ &\quad = \frac{3}{2}\int_0^r F(\rho)\rho\mathrm{d}\rho, \end{aligned}$$

  where

$$\begin{aligned} F(\rho) &= \int_0^{2\pi} \nabla G(X_0 + \rho\cos\theta, Y_0 + \rho\sin\theta) \cdot \begin{pmatrix} \rho\sin\theta \\ -\rho\cos\theta \end{pmatrix}\mathrm{d}\theta \\ &= -\int_{\mathcal{C}((X_0,Y_0),\rho)} \partial_X G\;\mathrm{d}X + \partial_Y G\;\mathrm{d}Y = 0 \qquad \text{by Green's theorem.} \end{aligned}$$

Finally, anticipating **Section 6.4**, the fact that the Wronskian of the five Abelian integrals is not uniformly 0 proves the desired linear independence. □

In the end, we have to consider linear combinations of five Abelian integrals, investigate the resulting sign alternations and apply a factor 4 (respectively 2) to every sign change detected on the small ovals (respectively big ovals).

$$\mathfrak{I}(h) = \alpha_{00}\mathfrak{I}_{00}(h) + \alpha_{20}\mathfrak{I}_{20}(h) + \alpha_{22}\mathfrak{I}_{22}(h) + \alpha_{40}\mathfrak{I}_{40}(h) + \alpha_{04}\mathfrak{I}_{04}(h),$$
$$\text{with} \quad \mathfrak{I}_{ij}(h) = -\int_{\Gamma(h)} \frac{x^i y^j}{y}\mathrm{d}x. \tag{6.10}$$

6.2.3 ▶ Formulas for the Abelian integrals $\mathfrak{I}_{ij}$

In order to evaluate the Abelian integrals $\mathfrak{I}_{ij}(h)$ for a given parameter h, we use explicit expressions using an appropriate parameterization of $\Gamma(h)$. For the small and big ovals, our parameterization consists in subdividing $\Gamma(h)$ into four parts, two of them being expressed under the form $y(x)$, and the two others under the form $x(y)$. **Sections 6.2.3** and **6.2.3** give the desired explicit integrals obtained with this parameterization, that will be used in **Section 6.3** for the rigorous computation of the $\mathfrak{I}_{ij}(h)$.

Note that the small ovals also admit a simple trigonometric parameterization (since $\Gamma(h)$ is a circle in the (x^2, y^2) plane) that we do not present here, as it can hardly be transposed to the rigorous evaluation on the big ovals.

In all this section, we assume $0 < X_0 < Y_0$.

■ Abelian integrals on the small ovals

We fix an energy level $h = r^2 \in (0, X_0^2)$, defining four symmetric *small ovals* as described in **Section 6.2.1**. Let $\Gamma(h)$ denote the one in the positive quadrant. By dividing the corresponding circle of center (X_0, Y_0) and radius r in the (x^2, y^2) plane into four parts (see **Figure 6.3a**), we get a parameterization of $\Gamma(h)$ as $\gamma_{1,h}^\circ \cup \gamma_{2,h}^\circ \cup \gamma_{3,h} \cup \gamma_{4,h}$ (see **Figure 6.3b**), where for a path $\gamma : [a, b] \to \mathbb{R}^2$, $\gamma^\circ : t \in [a, b] \mapsto \gamma(a + b - t) \in \mathbb{R}^2$ denotes the opposite path:

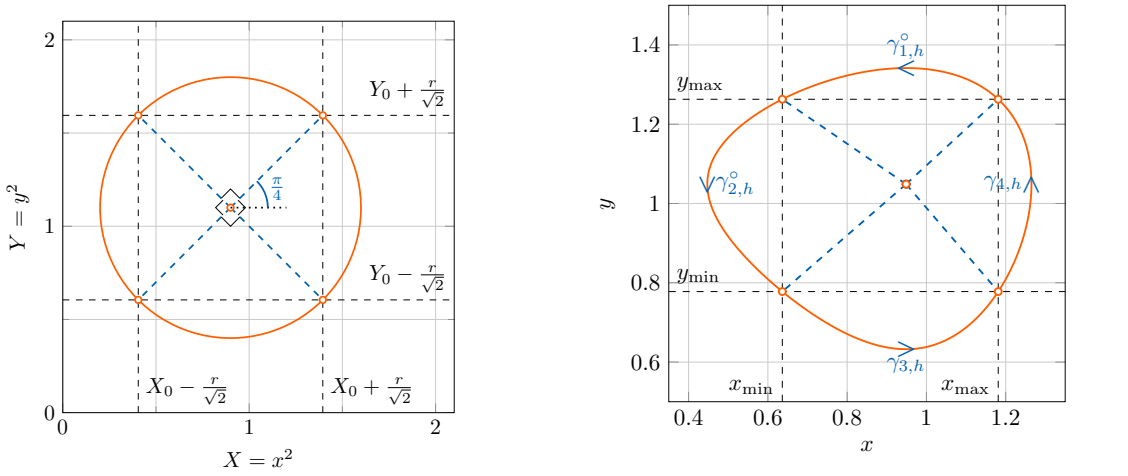
$$\gamma_{1,h} : x \in [x_{\min}, x_{\max}] \mapsto (x, y_{\text{up}}(x)), \qquad \gamma_{2,h} : y \in [y_{\min}, y_{\max}] \mapsto (x_{\text{left}}(y), y),$$
$$\gamma_{3,h} : x \in [x_{\min}, x_{\max}] \mapsto (x, y_{\text{down}}(x)), \quad \gamma_{4,h} : y \in [y_{\min}, y_{\max}] \mapsto (x_{\text{right}}(y), y).$$

The parametric expressions $y_{\text{up}}(x)$, $y_{\text{down}}(x)$, $x_{\text{left}}(y)$ and $x_{\text{right}}(y)$, as well as the corresponding path extremities $x_{\min}$, $x_{\max}$, $y_{\min}$ and $y_{\max}$, admit explicit expressions:

$$\begin{aligned}
x_{\min} &= \sqrt{X_0 - \frac{r}{\sqrt{2}}}, & x_{\max} &= \sqrt{X_0 + \frac{r}{\sqrt{2}}},\\
y_{\min} &= \sqrt{Y_0 - \frac{r}{\sqrt{2}}}, & y_{\max} &= \sqrt{Y_0 + \frac{r}{\sqrt{2}}},\\
\delta_y(x) &= \sqrt{r^2 - (x^2 - X_0)^2}, & \delta_x(y) &= \sqrt{r^2 - (y^2 - Y_0)^2},\\
y_{\text{down}}(x) &= \sqrt{Y_0 - \delta_y(x)}, & y_{\text{up}}(x) &= \sqrt{Y_0 + \delta_y(x)},\\
x_{\text{left}}(y) &= \sqrt{X_0 - \delta_x(y)}, & x_{\text{right}}(y) &= \sqrt{X_0 + \delta_x(y)},\\
x_{\text{left}}'(y) &= \frac{y(y^2 - Y_0)}{\delta_x(y) x_{\text{left}}(y)}, & x_{\text{right}}'(y) &= -\frac{y(y^2 - Y_0)}{\delta_x(y) x_{\text{right}}(y)}.
\end{aligned} \tag{6.11}$$

We can therefore express $\mathfrak{I}_{ij}(h)$ using integrals over segments of the real line:

$$\begin{aligned}\mathfrak{I}_{ij}(h) &= -\int_{\gamma^\circ_{1,h}} \frac{x^i y^j}{y}\,\mathrm{d}x - \int_{\gamma^\circ_{2,h}} \frac{x^i y^j}{y}\,\mathrm{d}x - \int_{\gamma_{3,h}} \frac{x^i y^j}{y}\,\mathrm{d}x - \int_{\gamma_{4,h}} \frac{x^i y^j}{y}\,\mathrm{d}x \\ &= \int_{x_{\min}}^{x_{\max}} x^i ({y_{\mathrm{up}}}^{j-1}(x) - {y_{\mathrm{down}}}^{j-1}(x))\,\mathrm{d}x + \int_{y_{\min}}^{y_{\max}} ({x_{\mathrm{left}}}^{i-1}(y) + {x_{\mathrm{right}}}^{i-1}(y)) y^j \frac{y^2 - Y_0}{\delta_x(y)}\,\mathrm{d}y. \end{aligned} \tag{6.12}$$



(a) Symmetric division of the circle

(b) Parameterization of $\Gamma(h)$ as $\gamma^\circ_{1,h} \cup \gamma^\circ_{2,h} \cup \gamma_{3,h} \cup \gamma_{4,h}$

Figure 6.3: Parameterization of small ovals ($0 < h < X_0^2$)

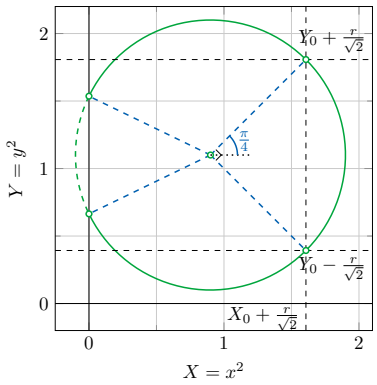
Abelian integrals on the big ovals

We now fix $h = r^2 \in (X_0^2, Y_0^2)$. Let $\Gamma(h)$ denote, among the two symmetric big ovals described in **Section 6.2.1**, the upper one located in the half plane $\{y > 0\}$. In the (x^2, y^2) plane, it is associated to the portion of the circle of center (X_0, Y_0) and radius r located in the positive quadrant. This arc is partitioned as in **Figure 6.4a**, giving the parameterization of $\Gamma(h)$ as $\gamma^\circ_{1,h} \cup \gamma^\circ_{2,h} \cup \gamma_{3,h} \cup \gamma_{4,h}$ (see **Figure 6.4b**):

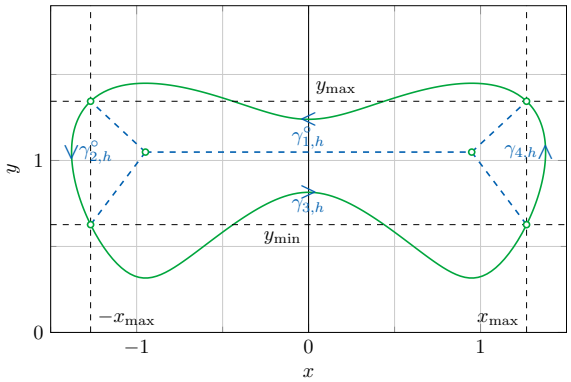
$$\begin{aligned} &\gamma_{1,h} : x \in [-x_{\max}, x_{\max}] \mapsto (x, y_{\mathrm{up}}(x)), \qquad \gamma_{2,h} : y \in [y_{\min}, y_{\max}] \mapsto (-x_{\mathrm{right}}(y), y), \\ &\gamma_{3,h} : x \in [-x_{\max}, x_{\max}] \mapsto (x, y_{\mathrm{down}}(x)), \quad \gamma_{4,h} : y \in [y_{\min}, y_{\max}] \mapsto (x_{\mathrm{right}}(y), y), \end{aligned}$$

where the parametric expressions $y_{\mathrm{up}}(x)$, $y_{\mathrm{down}}(x)$, $x_{\mathrm{left}}(y)$, $x_{\mathrm{right}}(y)$, and the path extremities $x_{\max}$, $y_{\min}$, $y_{\max}$ are still given by **(6.11)**. This allows us to give the following explicit expressions for $\mathfrak{I}_{ij}(h)$ on the big ovals:

$$\mathfrak{I}_{ij}(h) = 2\int_0^{x_{\max}} x^i ({y_{\mathrm{up}}}^{j-1}(x) - {y_{\mathrm{down}}}^{j-1}(x))\,\mathrm{d}x + 2\int_{y_{\min}}^{y_{\max}} {x_{\mathrm{right}}}^{i-1}(y) y^j \frac{y^2 - Y_0}{\delta_x(y)}\,\mathrm{d}y. \tag{6.13}$$



(a) Division of the portion of the circle in the positive quadrant



(b) Parameterization of $\Gamma(h)$ as $\gamma^\circ_{1,h} \cup \gamma^\circ_{2,h} \cup \gamma_{3,h} \cup \gamma_{4,h}$

■ **Figure 6.4**: Parameterization of big ovals ($X_0^2 < h < Y_0^2$)

6.3 Computer-assisted proof of the main theorem

The purpose of this section is to provide a rigorous computer-assisted proof of **Theorem 6.1** by choosing appropriate coefficients α_{ij} in Equation (6.10), computing rigorous interval enclosures for the evaluation of the Abelian integral $\mathfrak{I}(h)$, and counting the number of sign changes.

6.3.1 ▶ Computing the coefficients α_{ij}

Our strategy to determine these coefficients is the following. We first try to maximize the number of zeros on the small ovals domain $h \in (0, X_0^2)$, since by symmetry each sign change gives rise to four limit cycles. We uniformly discretize the interval $(0, X_0^2)$ with M points and evaluate the $\mathfrak{I}_{ij}(h)$ on that grid. At that point and due to efficiency reasons, we only perform numerical evaluations without error bounds. It is clear that for every subset of four points from the grid, there exists a non-trivial linear combination $\mathfrak{I}(h) = \sum \alpha_{ij}\mathfrak{I}_{ij}(h)$ of the five Abelian integrals $\mathfrak{I}_{ij}(h)$ that vanishes on these points. Hence, we repeat the process for all subsets of four points and count the resulting number of sign changes, hoping for an extra fifth one. We also count the number of sign changes obtained on the big ovals domain $h \in (X_0^2, Y_0^2)$, each of them inducing two limit cycles by symmetry. We finally consider the combination that maximizes the number of sign changes, counted with multiplicity induced by the symmetries.

This is how coefficients of **Table 6.1** were obtained, using a grid of size $M = 200$. The corresponding Abelian integral $\mathfrak{I}(h)$ shows 5 sign changes on the small ovals domain and 2 on the big ovals one, yielding the announced result of $5 \times 4 + 2 \times 2 = 24$ limit cycles. Note that these values should not be truncated, otherwise the expected sign alternations may not happen.

α_{00}	=	-0.78622148667854837664
α_{20}	=	0.87723523612653436051
α_{22}	=	1
α_{40}	=	0.23742713894293038223
α_{04}	=	-0.21823846173078863753

Table 6.1: Reference values for the coefficients α_{ij}, using a grid of size $M = 200$.

6.3.2 ▶ Rigorous pointwise evaluation of the Abelian integrals

We compute rigorous interval enclosures $\mathfrak{I}(h)$ of the values of Abelian integrals $\mathfrak{I}(h)$ for specific values of the energy level h, using Chebyshev models for the parameterizations and formulas established in Section 6.2.3. The existence of sufficiently many simple zeros needed to prove Theorem 6.1 is guaranteed by the sign alternation of these rigorous pointwise evaluations.

More specifically, for a fixed $r = \sqrt{h} \in (0, X_0)$, Chebyshev models are computed for the integrands appearing in Equation (6.12). All the needed operations are covered by Chapter 3. After that, those Chebyshev models are integrated over $[x_{\min}, x_{\max}]$ and $[y_{\min}, y_{\max}]$, and the results are summed to provide a rigorous enclosure of $\mathfrak{I}_{ij}(h)$. This operation is performed for each of the five Abelian integrals, yielding the expected enclosure $\mathfrak{I}(h)$ of $\mathfrak{I}(h)$, using Equation (6.10) and coefficients α_{ij} from Table 6.1.

We proceed similarly for $r = \sqrt{h} \in (X_0, Y_0)$, but using the parametric formulas for the big ovals (Equation (6.13)).

Table 6.2 gives interval enclosures $\mathfrak{I}(h)$ of $\mathfrak{I}(h)$ for 9 values of h using this rigorous pointwise evaluation method and the coefficients of Table 6.1. Figure 6.5 provides a graphical representation of the sign alternations on small and big ovals. All the computations were performed using the ChebValid C library[3] for rigorous numerics with Chebyshev models presented in Chapter 3.

Proof of Theorem 6.1. Let h_i $(1 \leqslant i \leqslant 9)$ denote the 9 values of h in Table 6.2, taken in increasing order. The intervals $\mathfrak{I}(h_i)$ given in this table are rigorous enclosures of the $\mathfrak{I}(h_i)$, by the correctness of the operations on Chebyshev models detailed in Chapter 3.

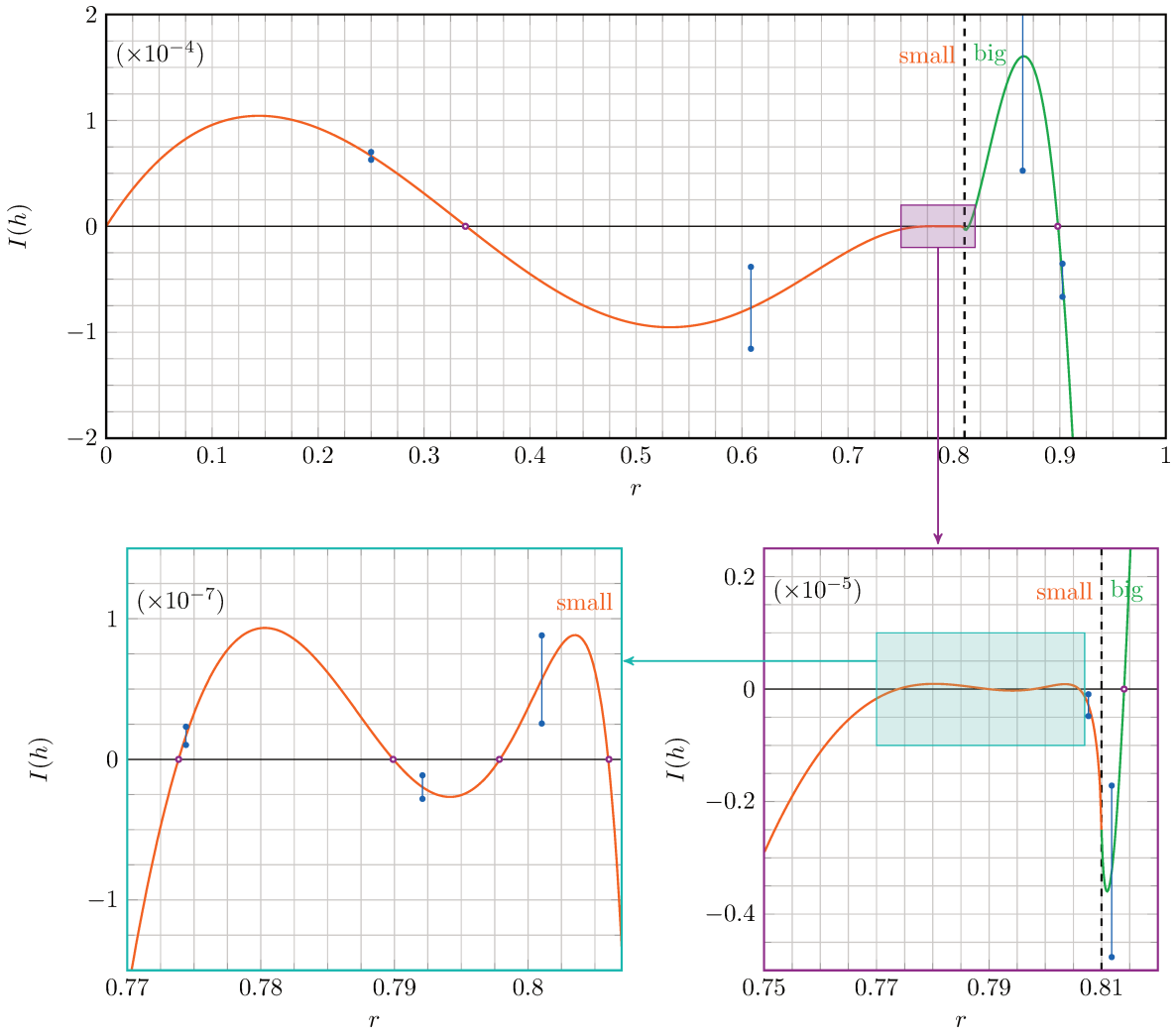
According to Theorem 6.3, there exists for each h_i, an $\varepsilon_i > 0$ such that $d(h_i, \varepsilon)$ and $\mathfrak{I}(h_i)$ share the same (strict) sign whenever $0 < \varepsilon \leqslant \varepsilon_i$. Hence, with $\varepsilon^* = \min_{0 \leqslant i \leqslant 9} \varepsilon_i > 0$, we have that $h \mapsto d(h, \varepsilon)$ alternates sign at least 5 times on $(0, X_0^2)$ and at least 2 times on (X_0^2, Y_0^2), for each fixed $0 < \varepsilon \leqslant \varepsilon^*$, giving respectively at least 5 and 2 isolated zeros in these intervals.

Finally, using the symmetries on the four small ovals and the two big ovals, we deduce the existence of at least $5 \times 4 + 2 \times 2 = 24$ limit cycles in the quartic system (6.8) whenever $0 < \varepsilon \leqslant \varepsilon^*$. □

Certified results

The results presented above were obtained using the C library ChebValid. However, as exemplified by the bug in the code used in [126], a higher level of confidence is not unnecessary.

[3] `https://gforge.inria.fr/projects/tchebyapprox/`



■ **Figure 6.5**: Abelian integral $\mathfrak{I}(h)$ on small and big ovals with the coefficients of **Table 6.1**. Values for the big ovals are halved to display a continuous plot. The blue vertical intervals are the rigorous enclosures of **Table 6.2**.

ovals	r	h	N	$\mathfrak{I}(h)^{[N]}$	$\mathfrak{I}(h)^{[N_{\text{ref}}]}$	sign($\mathfrak{I}(h)$)
small	0.5	0.25	15	[6.2827e-5,7.0092e-5]	[6.6457e-5,6.6458e-5]	+
	0.78	0.6084	15	[-1.1558e-4,-3.8299e-5]	[-7.6939e-5,-7.6938e-5]	−
	0.88	0.7744	70	[1.0236e-8,2.3226e-8]	[1.6730e-8,1.6731e-8]	+
	0.89	0.7921	100	[-2.8087e-8,-1.1346e-8]	[-1.9717e-8,-1.9716e-8]	−
	0.895	0.801025	135	[2.5476e-8,8.8149e-8]	[5.6812e-8,5.6813e-8]	+
	0.8987	0.80766169	250	[-4.8007e-7,-9.1649e-8]	[-2.9067e-7,-2.8104e-7]	−
big	0.901	0.811801	50	[-9.5281e-6,-3.4314e-6]	[-6.4798e-6,-6.4797e-6]	−
	0.93	0.8649	20	[1.0503e-4,5.3684e-4]	[3.2088e-4,3.2089e-4]	+
	0.95	0.9025	25	[-1.3308e-4,-7.0634e-5]	[-1.0186e-4,-1.0185e-4]	−

Table 6.2: Rigorous evaluation of $\mathfrak{I}(h)$ and sign alternations on the small and big ovals. The approximation degree N used in Chebyshev models is chosen to ensure that the resulting interval enclosure $\mathfrak{I}(h)^{[N]}$ guarantees the sign of $\mathfrak{I}(h)$. For comparison, precise enclosures $\mathfrak{I}(h)^{[N_{\text{ref}}]}$ with $N_{\text{ref}} = 300$ are also given.

To this aim, we also provide certified evaluations using the Coq library presented in **Chapter 3**. Only the computations for the small ovals, summarized in **Table 6.3** have been carried out at the moment of writing this manuscript, and the remaining ones are to be carried out in the near future.

r	N	p	time (s)	enclosure	sign
0.5	13	32	0.38	[1.56e-5,1.18e-4]	+
0.78	15	32	0.47	[-1.32e-4,-2.28e-5]	−
0.88	65	128	17.34	[1.83e-10,3.33e-8]	+
0.89	95	128	35.13	[-3.31e-8,-6.42e-9]	−
0.895	135	300	173.23	[3.94e-8,7.42e-8]	+
0.8987	250	350	596.66	[-3.86e-7,-1.86e-7]	−

Table 6.3: Certified enclosures of $\mathfrak{I}(r)$ for different values of r, computed with degree-N Chebyshev models and floating-point precision p.

6.4 Related ongoing works

When it comes to upper bounds for **Conjecture 6.2**, a relevant technique is that of verifying Chebyshev properties of certain families of Abelian integrals. Given a Hamiltonian H and (a family of) perturbations P and Q, we want to know when the associated collection of Abelian integrals form an *extended complete Chebyshev system* **[133]** (ECT-system for short). An ordered set of functions $(f_0, f_1, \ldots, f_n)$ forms an ECT-system on an interval J (the range of energies h in our case) if, for all $k = 1, 2, \ldots, n$, any nontrivial linear combination $\alpha_0 f_0(h) + \alpha_1 f_1(h) + \ldots + \alpha_k f_k(h)$ has at most k zeros on J counted with multiplicity.

For convenience, we state the precise definitions below, extending the Haar condition and the definition of Chebyshev systems (cf. **Definition 2.20**). Here J is a closed, non-empty interval of the real line.

Definition 6.5 *Let $f_0, f_1, \ldots, f_{n-1} \in \mathcal{C}^{n-1}(J)$.*

1. *The ordered set of functions $(f_0, f_1, \ldots, f_{n-1})$ is a* complete Chebyshev system *(in short, CT-system) on J if, for all $k = 1, 2, \ldots, n$, any nontrivial linear combination*

$$\alpha_0 f_0(x) + \alpha_1 f_1(x) + \ldots + \alpha_{k-1} f_{k-1}(x)$$

has at most $k-1$ zeros on J.

2. *The ordered set of functions $(f_0, f_1, \ldots, f_{n-1})$ is an* extended complete Chebyshev system *(in short, ECT-system) on J if, for all $k = 1, 2, \ldots, n$, any nontrivial linear combination*

$$\alpha_0 f_0(x) + \alpha_1 f_1(x) + \ldots + \alpha_{k-1} f_{k-1}(x)$$

has at most $k-1$ zeros on J counted with multiplicity.

It is clear that if $(f_0, f_1, \ldots, f_{n-1})$ is an ECT-system on J, then it is a CT-system on J.

Definition 6.6 *Let $f_0, f_1, \ldots, f_{k-1} \in \mathcal{C}^{k-1}(J)$. The* Wronskian *of $(f_0, f_1, \ldots, f_{k-1})$ at $x \in J$ is*

$$W[f_0, f_1, \ldots, f_{k-1}](x) = \det\left(f_j^{(i)}(x)\right)_{0 \leqslant i,j \leqslant k-1} = \begin{vmatrix} f_0(x) & f_1(x) & \ldots & f_{k-1}(x) \\ f_0'(x) & f_1'(x) & \ldots & f_{k-1}'(x) \\ \vdots & \vdots & \ldots & \vdots \\ f_0^{(k-1)}(x) & f_1^{(k-1)}(x) & \ldots & f_{k-1}^{(k-1)}(x) \end{vmatrix}.$$

Lemma 6.7 *Let $f_0, f_1, \ldots, f_{n-1} \in \mathcal{C}^{n-1}(J)$. The ordered set $(f_0, f_1, \ldots, f_{n-1})$ is an ECT-system on J if and only if for each $k = 1, 2, \ldots n$,*

$$W[f_0, f_1, \ldots, f_{k-1}](x) \neq 0 \text{ for all } x \in J.$$

Clearly, there is no hope that our five Abelian integrals are an ECT system, even on $h \in (0, X_0^2)$, since a combination with 5 zeros on that interval was given in the previous section. Instead, we consider the notion of Chebyshev system with positive accuracy [187].

Definition 6.8 *Let $f_0, f_1, \ldots, f_{n-1} \in \mathcal{C}^{n-1}(J)$, and $r \in \mathbb{N}$.*

1. *The ordered set of functions $(f_0, f_1, \ldots, f_{n-1})$ is a* complete Chebyshev system with accuracy r *on J if, for all $k \in [\![1, n]\!]$, any nontrivial linear combination*

$$\alpha_0 f_0(x) + \alpha_1 f_1(x) + \ldots + \alpha_{k-1} f_{k-1}(x)$$

has at most $k-1+r$ zeros on J.

2. *The ordered set of functions $(f_0, f_1, \ldots, f_{n-1})$ is an* extended complete Chebyshev system with accuracy r *on J if, for all $k \in [\![1, n]\!]$, any nontrivial linear combination*

$$\alpha_0 f_0(x) + \alpha_1 f_1(x) + \ldots + \alpha_{k-1} f_{k-1}(x)$$

has at most $k-1+r$ zeros on J counted with multiplicity.

Hence, if we prove that the five Abelian integrals are an ECT-system with accuracy 1, then we cannot hope to obtain more than five zeros on the small ovals domain $h \in (0, X_0^2)$. The theory of ECT systems is by far more complex than standard ECT systems [187]. Fortunately, for accuracy 1, we have the following simple characterization [187, Cor. 1.4].

Lemma 6.9 *Let $f_0, f_1, \dots, f_{n-1} \in \mathcal{C}^{n-1}(J)$. If the Wronskians $W[f_0, \dots, f_{k-1}]$ are non-vanishing over J for $k \in [\![1, n-1]\!]$, and the last Wronskian $W[f_0, \dots, f_{n-1}]$ has a single simple zero over J, then $(f_0, \dots, f_{n-1})$ is an ECT system with accuracy 1. More precisely:*

- *$(f_0, \dots, f_{n-2})$ is an ECT system;*
- *any nontrivial linear combination of $(f_0, \dots, f_{n-1})$ has at most n zeros (counted with multiplicity), and at least one reaches this bound.*

To be able to prove this property on the Wronskians, we cannot rely on the pointwise evaluation of Abelian integrals anymore. Instead, we need *continuous* rigorous representations for the Abelian integrals and their derivatives. Here is the road-map of our ongoing work:

1. Using Creative Telescoping techniques briefly presented in **Chapter 2**, we obtain LODEs with polynomial coefficients for the five Abelian integrals, as discussed in the next section.
2. Unfortunately, these equations are singular for $h = 0$ and $h = X_0^2$. For the left endpoint $h = 0$, one can however prove using the *Laplace transform* that the Abelian integrals are analytic around 0, and explicit truncated Taylor series can be computed. Although this is still ongoing work, this theoretically gives the possibility to compute rigorous Taylor models for the Abelian integrals (and their derivative) over some small interval $[0, \varepsilon]$.
3. By evaluating these Taylor modes at $h = \varepsilon$, one gets new initial conditions at ε. Hence, using the validation technique of **Chapter 4**, one can compute Chebyshev models for the Abelian integrals and their derivatives on the interval $[\varepsilon, X_0^2 - \varepsilon]$.
4. Finally, one would need rigorous representations over the last remaining part $[X_0^2 - \varepsilon, X_0^2)$. Unfortunately, the Abelian integrals are not analytic at X_0^2. However, computer algebra techniques show that they admit series expansions of the form:

$$a_0 + \sum_{n=1}^{+\infty} \left(a_n + b_n \log(X_0^2 - h)\right) (X_0^2 - h)^n.$$

   Hence, this example highly motivates the development of new validation techniques for rigorous approximations in non-polynomial bases.

A Creative Telescoping based approach

As mentioned above, one needs RPAs for the Abelian integrals and their derivatives in order to compute a RPA for the Wronskians and therefore bounding the number of zeros. One possibility is to represent the integrand of the Abelian integral with a *bivariate* rigorous polynomial approximation and afterwards to integrate it with respect to the integration variable. However, generic bivariate polynomial representations of degree N have size $O(N^2)$. Instead, we propose to obtain LODEs for the Abelian integrals, so that we can apply the validation technique of

Chapter 4. To this aim, we use Creative Telescoping techniques (see Section 2.1.2) over semi-algebraic sets.

As a preliminary remark, Green's theorem is used to rewrite the Abelian integral (6.10) as a two-dimensional integral over the interior of the oval:

$$\mathfrak{I}_{ij}(h) = \iint_{\mathrm{Int}(\Gamma(h))} (j-1)x^i y^{j-2} \mathrm{d}x\mathrm{d}y.$$

The integrand is a D-finite function of (x, y, h), with the following annihilator:

$$\mathfrak{Ann}_{ij} = \{\partial_h, x\partial_x - i, y\partial_y - (j-2)\}.$$

In order to be able to apply Creative Telescoping methods over the semi-algebraic set $\mathrm{Int}(\Gamma(h))$ which is not a Cartesian product, we consider the following approach, advocated in [188]. First, we rewrite the Abelian integral as:

$$\mathfrak{I}_{ij}(h) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (j-1)x^i y^{j-2} \mathbf{1}_{\mathrm{Int}(\Gamma(h))}(x, y) \mathrm{d}x\mathrm{d}y, \tag{6.14}$$

where $\mathbf{1}_{\mathrm{Int}(\Gamma(h)}(x, y) = 1$ iff $(x, y) \in \mathrm{Int}(\Gamma(h))$, 0 otherwise. We thus recover a regular Cartesian product with natural boundaries, since the integrand $(j-1)x^i y^{j-2}\mathbf{1}_{\mathrm{Int}(\Gamma(h))}(x, y)\mathrm{d}x\mathrm{d}y$ is zero outside the compact set $\mathrm{Int}(\Gamma(h))$. This integrand is no longer a smooth function, but can be seen as a *generalized function* or *distribution*, satisfying the following relations, obtained by multiplying those of $\mathfrak{Ann}_{ij}$ by $H(\mathbf{x}, \mathbf{y}) - \mathbf{h}$ to "regularize" what happens at the border $\Gamma(h)$:

$$\widetilde{\mathfrak{Ann}}_{ij} = \{(H(x, y) - h)\partial_h, (H(x, y) - h)(x\partial_x - i), (H(x, y) - h)(y\partial_y - (j-2))\}$$

The ideal $\widetilde{\mathfrak{Ann}}_{ij}$ must be first rewritten in a "canonical basis" (namely, as a non-commutative Gröbner basis). Afterwards, a Creative Telescoping algorithm, Takayama's algorithm [240], produces an annihilator for $\mathfrak{I}_{ij}(h)$, that is a LODE with polynomial coefficients in h. The main characteristics of the LODEs obtained for the five Abelian integrals are summarized in Table 6.4. The roots of the leading coefficient are the *singular points* of the differential equation, where the Picard-Lindelöf theorem does not apply. The points $h = 0, X_0^2, Y_0^2, X_0^2 + Y_0^2$ are simple zeros: they are called *regular singular* points. Their presence was predictible due to the geometry of the problem. We also notice other factors in the leading coefficients, but desingularization techniques show that they are *apparent singularities* that can be removed. Table 6.5 gives the main characteristics of the desingularized differential equations obtained for $X_0 = 9/10$ and $Y_0 = 11/10$.

Remark 6.10 *For the sake of simplicity, the presentation given here is rather intuitive. The rigorous framework for holonomic distributions and Creative Telescoping over semi-algebraic sets will be given in Chapter 9.*

Beside the explicit form of these LODEs, one also needs corresponding initial conditions in order to be able to compute RPAs. As explained in Step 2 of the road-map given before, we propose an approach based on Laplace transform.

Exponents		Characteristics of LODE	
i	j	order	leading coefficient
0	0	3	$16h(h-X_0)^2(h-Y_0)^2(h-X_0^2-Y_0^2)(-X_0^4+4hY_0^2-2X_0^2Y_0^2-Y_0^4)$
2	0	3	$-16h(h-X_0)^2(h-Y_0)^2(h-X_0^2-Y_0^2)(2h-X_0^2-Y_0^2)$
4	0	4	$-16h(h-X_0)^2(h-Y_0)^2(h-X_0^2-Y_0^2)(8hX_0^2-5X_0^4+4hY_0^2-6X_0^2Y_0^2-Y_0^4)$
2	2	4	$-16h(h-X_0)^2(h-Y_0)^2(h-X_0^2-Y_0^2)(4hX_0^2-3X_0^4-2X_0^2Y_0^2+Y_0^4)$
0	4	4	$16h(h-X_0)^2(h-Y_0)^2(h-X_0^2-Y_0^2)(X_0^4+4hY_0^2-2X_0^2Y_0^2-3Y_0^4)$

Table 6.4: Characteristics of LODEs obtained by Creative Telescoping.

Exponents		Characteristics of LODE	
i	j	order	leading coefficient
0	0	4	$404h(100h-81)(100h-121)(50h-101)$
2	0	4	$2h(100h-81)(100h-121)(50h-101)$
4	0	5	$404h(100h-81)(100h-121)(50h-101)$
2	2	5	$404h(100h-81)(100h-121)(50h-101)$
0	4	5	$404h(100h-81)(100h-121)(50h-101)$

Table 6.5: Characteristics of the desingularized LODEs with $X_0 = 9/10$ and $Y_0 = 11/10$.

Taylor expansions around 0 using the Laplace transform

Consider the Laplace transform of the integral (6.14):

$$\mathcal{L}(\mathfrak{I}_{ij})(s) = \int_0^\infty \mathfrak{I}_{ij}(h)e^{-sh}\mathrm{d}h.$$

When $\mathcal{L}(\mathfrak{I}_{ij})(s)$ is well-defined, one has

$$\mathcal{L}(\mathfrak{I}_{ij})(s) = \frac{j-1}{s}\int_{-\infty}^{\infty} x^i e^{-s(x^2-x_0)^2}\mathrm{d}x \int_{-\infty}^{\infty} y^{j-2}e^{-s(y^2-y_0)^2}\mathrm{d}y.$$

From this expression, one can prove that $\mathcal{L}(\mathfrak{I}_{ij})(s)$ has a closed-form in terms of Bessel functions and their derivatives. Specifically, let $I_\nu(x)$ (and $K_\nu(x)$) denote the modified Bessel function of first (and respectively second) kind. The following proposition holds.

Proposition 6.11 *Let*

$$L_\alpha(s) = \int_{-\infty}^{\infty} x^\alpha e^{-s(x^2-x_0)^2}\,dx, \tag{6.15}$$

which is defined for $\alpha > -1$. *Then*

$$L_0(s) = e^{-sx_0{}^2}(4s)^{-1/4}\tilde{L}_0(\sqrt{s}x_0),$$
$$L_2(s) = e^{-sx_0{}^2}(4s)^{-3/4}\left(\partial_a\tilde{L}_0\right)(\sqrt{s}x_0),$$
$$L_4(s) = e^{-sx_0{}^2}(4s)^{-5/4}\left(\partial_a^2\tilde{L}_0\right)(\sqrt{s}x_0),$$

where

$$\tilde{L}_0(a) = \mathrm{e}^{a^2/2}\sqrt{a}\left(\sqrt{2}\pi I_{1/4}\left(a^2/2\right) + K_{1/4}\left(a^2/2\right)\right). \tag{6.16}$$

Proof. Equation (6.15) becomes, with the change of variable $(4s)^{1/4}x = z$:

$$L_\alpha(s) = e^{-sx_0{}^2}(4s)^{-(\alpha+1)/4}\int_{-\infty}^{\infty} z^\alpha e^{-z^4/4+s^{1/2}x_0z^2}\mathrm{d}z.$$

Consider now, for $a \geqslant 0$ and $\alpha > -1$, the parametric integral:

$$\tilde{L}_\alpha(a) = \int_{-\infty}^{\infty} z^\alpha e^{-z^4/4+az^2}\mathrm{d}z,$$

which satisfies:

$$\partial_a^2\tilde{L}_\alpha - 2a\partial_a\tilde{L}_\alpha - (\alpha+1)\tilde{L}_\alpha = 0. \tag{6.17}$$

This can be easily seen by by-parts integration:

$$\partial_a^2\tilde{L}_\alpha = \int_{-\infty}^{\infty} z^{\alpha+4}e^{-z^4/4+az^2}\mathrm{d}z = \int_{-\infty}^{\infty} -z^{\alpha+1}(-z^3+2az)e^{-z^4/4+az^2}\mathrm{d}z + \int_{-\infty}^{\infty} 2az^{\alpha+2}e^{-z^4/4+az^2}\mathrm{d}z.$$

Moreover for $\alpha = 0$, one has that $\tilde{L}_0(0) = \frac{\pi}{\gamma(3/4)}$ and $\partial_a\tilde{L}_0(0) = \sqrt{2}\gamma(3/4)$, where we denote the Gamma function by γ. One can check that the solution of Equation (6.17) is given by (6.16).

□

We also need this crucial lemma, sometimes refered to as *converse Watson's lemma* [262, Chap. I.5].

Lemma 6.12 (Converse Watson's lemma) *Let* f *be continuous over* $[0,+\infty]$, *such that its Laplace transform* F *is well-defined for* $\mathrm{Re}(s) > s_0 \in \mathbb{R}$:

$$F(s) := \mathcal{L}(f)(s) = \int_0^{+\infty} f(t)e^{-st}\mathrm{d}t, \qquad \mathrm{Re}(s) > s_0.$$

If F *admits the following asymptotic series expansion for* $s \to +\infty$:

$$F(s) \sim \sum_{n=0}^{+\infty} \frac{a_n}{n!\ s^{n+1}}, \qquad s \to +\infty,$$

then f *admits the following asymptotic expansion at* 0

$$f(t) \sim \sum_{n=0}^{+\infty} a_n t^n, \qquad t \to 0^+.$$

This lemma therefore shows that the Abelian integrals have a power series asymptotics around 0. Combining this result with the above section, stating that 0 is a regular singularity, we get that the Abelian integrals are *analytic* at 0.

Remark 6.13 *Note however that for Abelian integrals involving* negative powers *of y, i.e., $\mathfrak{I}_{00}$, $\mathfrak{I}_{20}$ and $\mathfrak{I}_{40}$, cannot be directly proved to be analytic with* ***Lemma 6.12****. Instead, we proceed by analytic continuation with respect to the exponent of y in the integral.*

Now, the first terms of the Taylor series can be explicited computed from the expressions of the Laplace transform (**Proposition 6.11**), using **Lemma 6.12**. The differential equations obtained in the previous section, translated into recurrences on the Taylor coefficients, can be used to efficiently compute the truncated Taylor series to very high degree. Finally, we plan to use elementary bounding techniques for recurrences to bound the remainder. In the end, we will obtain Taylor models for the Abelian integrals and their derivatives.

Validated Trajectories for Spacecraft Proximity Operations

7

– Cum, n-ai auzit că s-au lansat?
– În orbită sau ce câcat?
— Grasu XXL, Turbofin

As mentioned in the introduction of this manuscript, a wide range of applications in the domain of safety-critical engineering may benefit from rigorous numerics and other validated techniques. This is particularly true for many aerospace problems involving long-term integration of ODEs. Conversely, these applications provide interesting "real-life" problems for the rigorous numerics community. This chapter relates a joint work with two other PhD students in my research group at the LAAS laboratory, Paulo Ricardo Arantes Gilz [5] and Clément Gazzino [89]. Their works target the design of control laws for aerospace problems, more specifically spacecraft rendezvous for the former, and satellite station keeping for the latter. The purpose of this collaboration is to apply the validated techniques for ODEs using RPAs of **Chapters 4** and **5**, to design rigorous tools for the above mentioned aerospace problems. This work gave rise to an article "Validated Semi-Analytical Transition Matrix for Linearized Relative Spacecraft Dynamics via Chebyshev Polynomials", published in the proceedings of the *2018 Space Flight Mechanics Meeting*, organized by the *American Institute of Aeronautics and Astronautics* (AIAA).

7.1 Introduction and related works

During guidance and control procedures of orbiting spacecraft, the respect of positioning and space constraints is decisive for successful missions achievement. The development of algorithms capable of fulfilling these constraints is directly related to how precisely the spacecraft trajectories are known. Since accuracy is essential for these procedures, the prevention and

estimation of errors arising from approximations and numerical computations become critical.

For spacecraft proximity operations (spacecraft rendezvous, station keeping, collision avoidance), the relative dynamics are often linearized for both propagation or control purposes. More specifically, when the magnitude of the relative motion of the spacecraft is small compared to its distance to the Earth, one linearizes the equations of motion, which implies solving simpler linear differential equations (LODEs), that is, computing the *state transition matrix* (STM) representing the basis of solutions for the LODE. Seminal works proposed closed-form solutions (in a broader sense to be specified) for simple models, e.g. **[267]** for linearized Keplerian relative motion **[245]**. Nowadays, such "closed-form" STM solutions are known for a wider range of linearized models, including relevant perturbations in eccentric orbits about Earth and other bodies (see **[237]** for a detailed survey on that topic). However, most of these solutions are not "closed forms" in the strong meaning used throughout this thesis, requiring for example to solve implicit equations, such as the Kepler equation. This therefore does not provide a concrete continuous representation of the solution.

Alternatively, and in some cases where the perturbating forces are given by measurements or ephemerides, propagation can be performed with numerical iterative schemes (like Euler or Runge-Kutta). The main drawback of this discretization approach is that the number of needed evaluation points can be prohibitive and the discretization error is difficult to estimate precisely. Moreover, for control laws design purposes, analytical solutions are preferable, since various constraints (such as saturation, restricted space regions, etc.) need to be satisfied on continuous time domains and not only on discretization grids.

In this context, we consider solving linear ordinary differential equations via rigorous polynomial approximations in Chebyshev series (cf. **Chapters 4** and **5**), to get validated transition matrices describing the evolution of spacecraft trajectories.

This approach is applied to the study of two heterogeneous examples, thus highlighting its generality:

- First, we consider the linearized impulsive rendezvous framework, demonstrating how to use RPAs to provide a validated propagation of the relative dynamics between spacecraft. This is then exploited for the hovering phases of the spacecraft rendezvous, where we conceive a validated model predictive control based on semi-definite programs.

- Second, we propose a semi-analytical transition matrix, for a simplified model of geostationary orbits. Specifically, the dynamics, expressed in terms of the relative equinoctial orbital elements, are linearized taking into account the J2 Earth oblateness effect, that is, the perturbation of the Keplerian dynamics due to the Earth's non-sphericity at its poles.

For completeness, we provide in what follows a very brief general context for each of these examples.

7.1.1 ▶ Linearized impulsive rendezvous and model predictive control

Since the first space missions involving more than one vehicle, space rendezvous between two spacecraft has become a key technology raising relevant control issues. These missions consist

in the execution of a sequence of maneuvers aiming to bring a spacecraft (referred to as chaser) to the vicinity of a passive target (e.g. International Space Station), whose trajectory evolves around a central body. Since the '60s, many ideas have been developed, and today, we are interested in successful RdV which minimizes fuel consumption, with increased autonomy (no human operator). This implies that validation of computations and solutions is at stake. Impulsive RdV problems concern in practice a large number of satellites, which are equipped with ergol thrusters. The impulsive approximation for the thrust means that an instantaneous velocity increment is applied to the chaser for each impulse. This is relevant because the chaser's chemical engines provide high level of thrust during a short time with respect to the target orbital period, which leads to an extremely rapid change of velocity, which can thus be modeled as a jump at firing time.

Many model predictive control (MPC) algorithms have been developed since the 90's for spacecraft rendezvous (see for instance **[104, 47]**), since they provide fuel-efficient and *flyable* control solutions. The MPC uses *a priori* knowledge about the dynamics of the relative motion between spacecraft to iteratively compute the control corrections that fulfill fuel-optimality and various and uncertainties. These are directly accounted for in the trajectory design by formulating and solving a constrained optimal control problem which is repeatedly solved. Either dynamics discretization **[214, 241]** or analytic transitions matrices **[121, 8, 7]** were used for solving this problem. As previously mentioned, in this work we consider transition matrices approximated by RPAs.

This is very pertinent, since so far, several recent works took advantage of non-rigorous polynomial approximations – sometimes qualified as *semi-analytical methods* in the control theory community – in the context of model predictive control (MPC) and optimal impulsive constrained control **[70, 7]**. Their works follow the general framework of semidefinite programming (SDP) based on nonnegative polynomials written as sums of squares (SOS) **[182]**. From the numerical point of view, the efficiency of spectral methods with Chebyshev expansions was highlighted by recent works in the context of orbital mechanics **[211]**. They started to successfully replace the classical Taylor series-based algebra for intrusive approaches, which has already many applications to astrodynamics and optimal control for proximity operations **[158, 159, 71]**.

However, the scope of our work is not limited to numerical efficiency, since we also provide rigorous error bounds for the entries of the transition matrix, which are given as Chebyshev truncated series. This is particularly useful in optimization algorithms for optimal control where a trade-off must be done between low-degree polynomials for efficiency and accurate results, as it will be further shown in **Section 7.2**.

7.1.2 ▶ Station keeping on Geostationary Earth Orbits

Telecommunication satellites on a Geostationary Earth Orbit (GEO) have to stay above a fixed point of the Earth, at a position called station keeping position at zero latitude and at a given longitude. However, the geostationary orbit is computed assuming that the Earth produces a central gravitational attraction force. Other forces act on the satellite, such as the zonal and tesseral terms of gravitational potential of the Earth, the Sun and the Moon attractions and the Sun radiation pressure. It is therefore mandatory to control the spacecraft trajectory so that it remains in the vicinity of the station keeping point.

Some semi-analytical nonlinear models like the CNES Orange model [46] may be used to describe precisely the evolution of a geostationary Earth orbit affected by these orbital perturbations. However, when dealing with station keeping objectives, in most cases, a simpler linearized model, describing the spacecraft relative motion dynamics, has to be derived for the design of the station keeping strategy. In this linear, possibly time-varying setting, and after the seminal works of Hill-Clohessy-Wiltshire and Tschauner-Hempel [245], a very large number of contributions are have been proposed towards new allegedly closed-form *state transition matrix* (STM), depending on the orbital perturbations included in the linearized model, on the state representation used and on the linearizing assumptions (see [237] for a recent extensive survey on this topic).

Among the orbital perturbations relevant in a geostationary context, even if the J2 disturbing effect of the oblateness of the Earth is not the dominant one, it has a nonnegligible effect on the direction of the inclination vector drift [233, Sec. 4.4], and on the value of the true geostationary radius [230]. Hence, the study of solutions of the equation of the orbital motion perturbed by J2 remains of great interest. In this setting, closed form (in a broader sense) STM solutions are proposed in [91] using equinoctial elements. Our approach is complementary and consists in using the tools developed and implemented in **Chapters 4** and **5** to compute *polynomial approximate* STM together with *rigorous* error bounds. The fact that polynomials can be very efficiently evaluated using floating-point arithmetic makes such an approach competitive against closed-form but possibly complicated solutions.

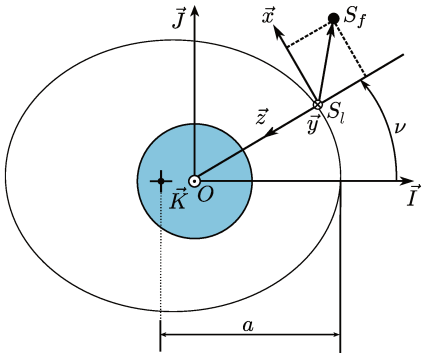
The work presented in **Section 7.3** is still experimental, and focuses on the validation of polynomial approximate STMs with respect to the linearized model. This is a first step towards further use of rigorous numerics in the station keeping problem: validation of these STMs with respect to the nonlinear equations of motion, efficient design of a station keeping strategy using these polynomial STMs, etc.

7.2 Validated linearized impulsive rendezvous and model predictive control

As a first example of an aerospace application of the validation method for LODEs presented in **Chapters 4** and **5**, we consider the impulsive spacecraft rendezvous problem previously described. The capacity of developing algorithms to address this problem is directly related to how precisely the trajectories developed by each spacecraft involved in the mission are known and, given that precision is a must for these procedures, the prevention and estimation of errors arising from approximations and numerical computations become a critical subject in this context.

Hereafter we describe the mathematical frameworks adopted to model the rendezvous problem. In **Figure 7.1**, the frames used to model the relative motion between the leader S_l and the follower S_f spacecraft are depicted. The Earth-Centered Inertial (ECI) frame is given by $\left\{O, \overrightarrow{I}, \overrightarrow{J}, \overrightarrow{K}\right\}$. The moving Local Vertical / Local Horizontal (LVLH) frame is centered on

the leader spacecraft at S_l and given by $\{S_l, \vec{x}, \vec{y}, \vec{z}\}$ (see [81] for details). Under Keplerian assumptions, the leader orbit is mainly described by its semi-major axis a and its eccentricity $0 < e < 1$. Then, the leader is located on its orbit by the true anomaly ν[1].



■ **Figure 7.1**: Inertial and relative frames

The relative motion is defined as the time history of the relative vector $\overrightarrow{S_l S_f}$. This motion has a state space representation with the following state vector:

$$X(t) = [x(t),\ y(t),\ z(t),\ \dot{x}(t),\ \dot{y}(t),\ \dot{z}(t)]^T.$$

Assuming the relative navigation hypothesis, $\|\overrightarrow{S_l S_f}\| \ll \|\overrightarrow{OS_l}\|$, a linear time-variant (LTV) representation for the relative dynamics is obtained: $\dot{X}(t) = A(t)X(t)$ [244]. By performing the state similar transformation:

$$\bar{X}(\nu) = \mathcal{T}(\nu)X(t), \quad \mathcal{T}(\nu) = \begin{bmatrix} (1 + e\cos\nu)\mathbf{1}_3 & \mathbf{0}_3 \\ -e\sin\nu\,\mathbf{1}_3 & \sqrt{\frac{a^3(1-e^2)^3}{\mu(1+e\cos\nu)^2}}\mathbf{1}_3 \end{bmatrix}, \tag{7.1}$$

where $\bar{X}(\nu) = [\bar{x}(\nu), \bar{y}(\nu), \bar{z}(\nu), \bar{x}'(\nu), \bar{y}'(\nu), \bar{z}'(\nu)]^T$ and μ is Earth's gravitational constant, the simplified linearized Tschauner-Hempel equations are obtained:

$$\bar{x}''(\nu) = 2\bar{z}'(\nu), \quad \bar{y}''(\nu) = -\bar{y}'(\nu), \quad \bar{z}''(\nu) = \frac{3}{1 + e\cos(\nu)}\bar{z}(\nu) - 2\bar{x}'(\nu), \tag{7.2}$$

where $(\cdot)' = \frac{d(\cdot)}{d\nu}$ and $(\cdot)'' = \frac{d^2(\cdot)}{d\nu^2}$. We observe that the in-plane motion ($\bar{x}$ and $\bar{z}$) is decoupled from the out-of-plane motion ($\bar{y}$). The latter involves only one coordinate and is a simple harmonic oscillator. The former involves two coordinates, but $\bar{x}'$ can be easily eliminated from the equation of $\bar{z}$, which becomes:

$$\bar{z}''(\nu) + \left(4 - \frac{3}{1 + e\cos\nu}\right)\bar{z}(\nu) = c, \tag{7.3}$$

where the constant c is defined from the initial conditions:

$$c = 4\bar{z}(\nu_0) - 2\bar{x}'(\nu_0).$$

[1]The true anomaly is an angular parameter that defines the position of a body moving along a Keplerian orbit. It is in a one-to-one correspondence with time. For the sake of brevity, these variables are interchanged without further remarks in this work.

First, the validation method for LODEs, presented in Chapter 4 and implemented in our ChebValid C library[2], is applied on the Tschauner-Hempel equations in Section 7.2.1. This provides RPAs for the spacecraft trajectory, that is, approximating polynomials in Chebyshev basis with validated error bounds. Then, in Section 7.2.2, we develop a model predictive control algorithm for spacecraft rendezvous hovering phases [7]. The underlying polynomial optimization problem justifies the need for low-degree but still accurate polynomial approximations of the transition matrix. Finally, we use the power of the validation method to perform an *a posteriori* verification of an instance of rendezvous by providing a rigorous enclosure of the chaser's final position. This is particularly important for safety critical missions.

7.2.1 ▶ Rigorous and semi-analytical transition matrix for Tschauner-Hempel equations

Before running the method on LODE (7.3), we first rescale the interval $[\nu_0, \nu_f]$ to $[-1, 1]$ by introducing the independent variable $\tau \in [-1, 1]$ and letting $\nu(\tau) = \nu_0(1-\tau)/2 + \nu_f(1+\tau)/2 = \omega\tau + \theta$ with $\omega = (\nu_f - \nu_0)/2$, $\theta = (\nu_0 + \nu_f)/2$, and $Z(\tau) = z(\nu(\tau))$. We obtain:

$$Z''(\tau) + \omega^2 \left(4 - \frac{3}{1 + e\cos\nu(\tau)}\right) Z(\tau) = \omega^2 c, \tag{7.4}$$

together with rescaled initial conditions:

$$Z(-1) = z(\nu_0), \qquad Z'(-1) = \omega z'(\nu_0).$$

In particular, we observe that the magnitude of the coefficients in Equation (7.4) grows quadratically with the length of the interval $[\nu_0, \nu_f]$ over which we want to approximate the trajectory.

Since the coefficient $\alpha(\tau)$ of Equation (7.4):

$$\alpha(\tau) := 4 - \frac{3}{1 + e\cos\nu(\tau)},$$

is not polynomial, we must provide a rigorous polynomial approximation for it. The cosine function $\tau \mapsto \cos\nu(\tau)$ is approximated by applying our validation method to the harmonic oscillator differential equation:

$$y''(\tau) + \omega^2 y(\tau) = 0, \qquad y(-1) = \cos\nu_0, \quad y'(-1) = -\omega\sin\nu_0. \tag{7.5}$$

Now, using the elementary operations defined in Chapter 3, we get a RPA for $\alpha(\tau)$. Figure 7.2a shows the evolution of the minimal degree p needed to approximate the coefficient $\tau \mapsto \omega^2(4 - 3/(1 + e\cos\nu(\tau)))$ within a $\mathcal{W}^1$-error less than 1, in function of the eccentricity e and the total time interval $[\nu_0, \nu_f]$.

Integral transform and numerical solving

Following the integral transform technique described in Chapter 4, we let $\varphi(\tau) = Z''(\tau)$, so that $Z(\tau)$ now becomes:

$$Z(\tau) = Z(-1) + (\tau+1)Z'(-1) + \int_{-1}^{\tau}\int_{-1}^{s} \varphi(u)\mathrm{d}u = Z(-1) + (\tau+1)Z'(-1) + \int_{-1}^{\tau}(\tau - s)\varphi(s)\mathrm{d}s.$$

[2] available at `https://gforge.inria.fr/projects/tchebyapprox/`

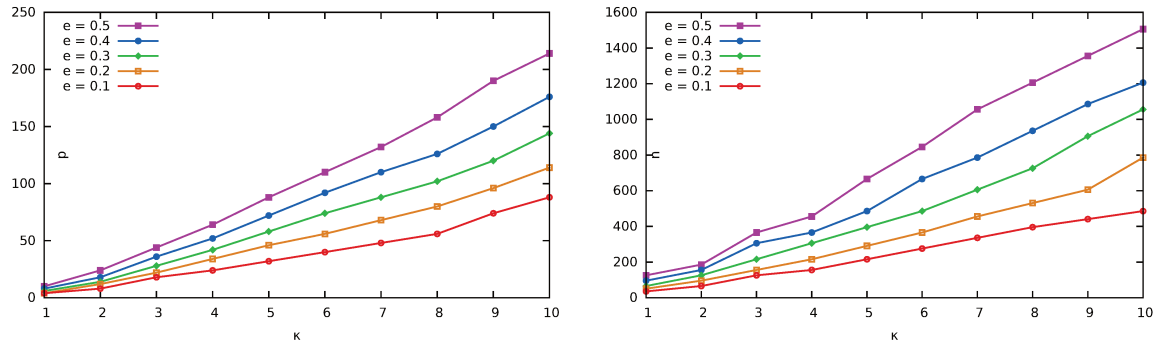
We thus obtain the integral equation:

$$\varphi(\tau) + \int_{-1}^{\tau} \alpha(\tau)(\tau - s)\varphi(s)\mathrm{d}s = \omega^2 c - \alpha(\tau)(Z(-1) + (\tau + 1)Z'(-1)),$$

where $\alpha(\tau) = 4 - 3/(1 + e\cos\nu(\tau))$. For the numerical solving, $\alpha(\tau)$ is firstly replaced by a polynomial approximation $a(\tau)$. Then, proceeding as in **Chapter 4**, the resulting infinite-dimensional equations are truncated at a chosen index N_{app}. Finally, the resulting almost-banded system is solved using the algorithms of **[191]** (recalled in **Chapter 4** under the names OlverTownsendQR and OlverTownsendBackSubs, so as to obtain a degree N_{app} polynomial approximation φ° of the solution.

Validation

The C code library ChebValid is used to validate the numerical solution φ°. The timings strongly depend on the minimum value for the truncation index N_{val} that the method finds and which ensures that the obtained operator is contracting. **Figure 7.2b** gives these values in function of the time interval $\nu_f - \nu_0$ and the eccentricity e of the target reference orbit. We stress out the fact that these values only depend on the equation (that is, ν_0, ν_f and e) and not on the degree of the candidate approximate solution φ°, since the contracting operator $\mathbf{T}$ is completely independent of this approximation.



(a) Approximation degree p needed to approximate coefficient $\tau \mapsto \omega^2(4 - 3/(1 + e\cos\nu(\tau)))$ with a $\mathcal{U}^1$-error less than 1.

(b) Truncation order N_{val} needed to obtain a contracting Newton-like operator for LODE (7.4).

Figure 7.2: Parameters evolution during validation of LODE (7.4) in function of eccentricity e and total time $[\nu_0, \nu_f] = [0, 2\kappa\pi]$.

Figure 7.2b shows that a straightforward application of the validation method tends to rapidly become time-consuming after about 10 periods. In the following section, we explain how to exploit the periodicity of the equation to validate a freely propagated trajectory over a large number of periods.

Long-term validated integration techniques

Since Equation (7.3) is linear and 2π-periodic, validating a transition matrix over $[\nu_0, \nu_0 + 2\pi]$ is a good starting point for most applications. Let

$$\Phi(\nu, \nu_0) = \left(\begin{array}{c|c|c|c} \bar{x}_{(i)}(\nu) & \bar{x}_{(ii)}(\nu) & \bar{x}_{(iii)}(\nu) & \bar{x}_{(iv)}(\nu) \\ \bar{z}_{(i)}(\nu) & \bar{z}_{(ii)}(\nu) & \bar{z}_{(iii)}(\nu) & \bar{z}_{(iv)}(\nu) \\ \bar{x}'_{(i)}(\nu) & \bar{x}'_{(ii)}(\nu) & \bar{x}'_{(iii)}(\nu) & \bar{x}'_{(iv)}(\nu) \\ \bar{z}'_{(i)}(\nu) & \bar{z}'_{(ii)}(\nu) & \bar{z}'_{(iii)}(\nu) & \bar{z}'_{(iv)}(\nu) \end{array}\right),$$

where the column of index (i), (resp. (ii), (iii) and (iv)) is a validated approximation of the in-plane trajectory corresponding to the initial conditions $\bar{x}(\nu_0) = 1$ (resp. $\bar{z}(\nu_0) = 1$, $\bar{x}'(\nu_0) = 1$ and $\bar{z}'(\nu_0) = 1$), all the other initial values being set to 0. Each entry is given as a Chebyshev model computed by ChebValid.

To ensure a validated propagation over several periods, the trajectory can be approximated by rigorous piecewise polynomial approximations over each period $[\nu_0 + 2k\pi, \nu_0 + 2(k+1)\pi]$:

$$\begin{pmatrix} \bar{x}(\nu) \\ \bar{z}(\nu) \\ \bar{x}'(\nu) \\ \bar{z}'(\nu) \end{pmatrix} \in \Phi(\nu - 2k\pi, \nu_0)\mathbb{J}^k \begin{pmatrix} \bar{x}(\nu_0) \\ \bar{z}(\nu_0) \\ \bar{x}'(\nu_0) \\ \bar{z}'(\nu_0) \end{pmatrix}, \tag{7.6}$$

where $\mathbb{J} = \Phi(\nu_0 + 2\pi, \nu_0)$ is the interval matrix of rigorous enclosures of the final states after one period. Let us remark that this method does not provide a uniform rigorous polynomial approximation over the whole time interval under consideration. Moreover, if the entries of $\mathbb{J}$ are rather loose intervals, which occurs when $\Phi(\cdot, \nu_0)$ is made of low-degree polynomial approximations, then the intervals in $\mathbb{J}^k$ will rapidly become very large and all precision is lost after a certain number of periods.

In a second step, one can obtain a certified uniform polynomial approximation over the whole time interval if the entries of $\mathbb{J}^k$ are sufficiently tight over the required k periods. For that, one uses a numerical polynomial approximation for the trajectory $\bar{X}_{xz}(\nu) = (\bar{x}(\nu), \bar{z}(\nu), \bar{x}'(\nu), \bar{z}'(\nu))$, over the whole interval $[\nu_0, \nu_f]$, where $\nu_f = \nu_0 + 2\kappa\pi$. This is validated *a posteriori* by bounding the difference between (7.6) and the candidate approximation $\bar{X}_{xz}(\nu)$, both considered over each period $[\nu_0 + 2k\pi, \nu_0 + 2(k+1)\pi]$ $(0 \leqslant k < \kappa)$. To restrict $\bar{X}_{xz}(\nu)$ to a period $[\nu_0 + 2k\pi, \nu_0 + 2(k+1)\pi]$ $(0 \leqslant k < \kappa)$, note that initially, our method provides a truncated Chebyshev series which is rescaled such that its definition interval is $[-1, 1]$. Hence, for each period, $\bar{X}_{xz}$ must be composed on the right by an affine time rescaling as to extract the desired time subinterval and compare the resulting Chebyshev truncated series with the corresponding precise piecewise approximation. Finally, the sum between the bound of this difference in the $\mathcal{U}^1$-norm and the rigorous error bound of the precise solution gives a safe overestimation of the uniform error.

7.2.2 ▶ Model predictive control for the rendezvous hovering phases using RPA-based transition matrices

Hereafter we focus on the study of the hovering phases of the orbital spacecraft rendezvous missions. The hovering phases are the stages at which the follower satellite is required to remain

in the interior of a delimited zone of the space with respect to the target spacecraft [123]. The idea is to provide a validated model predictive control (MPC) algorithm to steer the follower satellite in a fuel-optimal way to the hovering region. The MPC uses *a priori* knowledge about the dynamics of the relative motion between spacecraft to iteratively compute the control corrections that fulfill fuel-optimality and constraints [8, 7].

Validated relative dynamics

We consider the previously described states $X(t)$, $\bar{X}(\nu)$ and the relative dynamics given by the simplified linearized Tschauner-Hempel equations in (7.2). For a given time t (conversely, a true anomaly value ν), the state right after an impulsive velocity correction $\Delta V = [\Delta V_x, \Delta V_y, \Delta V_z]^T \in \mathbb{R}^3$ is defined as $X^+(t)$ and can be computed by:

$$X^+(t) = X(t) + B\,\Delta V(t),\ B = [\mathbf{0}_3\ \mathbf{1}_3]^T.$$

Performing the variable changes $X(t) \overset{(7.1)}{\rightarrow} \bar{X}(\nu)$, the state after the impulse is given by:

$$\bar{X}^+(\nu) = \bar{X}(\nu) + \bar{B}(\nu)\,\Delta V(\nu),\ \bar{B}(\nu) = \mathcal{T}(\nu)\,B.$$

Let $\bar{\Phi}(\nu_f, \nu_0)$ be the exact transition matrix of the system of equations (7.2), from an initial ν_0 to a final ν_f. By considering N impulsive velocities corrections applied at $\nu_1 < \ldots < \nu_N$, the propagation of the state can be formulated as:

$$\bar{X}^+(\nu_N) = \bar{\Phi}(\nu_N, \nu_1)\bar{X}(\nu_1) + \textstyle\sum_{k=1}^{N} \bar{\Phi}(\nu_N, \nu_k)\bar{B}(\nu_k)\Delta V_k.$$

By applying the Chebyshev series approximation method previously presented, one can obtain rigorous polynomial approximations $\Phi(\nu, \nu_0)$ on an interval $[\nu_0, \nu_f]$, that is approximations Φ°_{ij} and error bounds ε_{ij} satisfying:

$$|\Phi^\circ_{ij}(\nu_0, \nu) - \bar{\Phi}_{ij}(\nu_0, \nu)| \leqslant \varepsilon_{ij}, \quad \forall \nu \in [\nu_0, \nu_f].$$

Then, the propagation of the relative dynamics for $\nu \in [\nu_N, +\infty)$ can be represented by the state $\mathbb{X}(\nu)$, which provides a rigorous approximation of $\bar{X}(\nu)$:

$$\mathbb{X}(\nu) = \Phi(\nu, \nu_1)\bar{X}(\nu_1) + \sum_{k=1}^{N} \Phi(\nu_N, \nu_k)\bar{B}(\nu_k)\Delta V_k.$$

Propellers, fuel-consumption and saturation

We assume that the follower spacecraft has six identical propellers, one pair symmetrically and oppositely disposed by axis. The fuel consumption is then modeled by the sum of the absolute value of the thrusts applied in each direction:

$$\mathcal{J}(\Delta V) = \sum_{i=1}^{N} \|\Delta V(\nu_i)\|_1 = \sum_{i=1}^{N} |\Delta V_x(\nu_i)| + |\Delta V_y(\nu_i)| + |\Delta V_z(\nu_i)|.$$

Assuming that the saturation threshold for each propeller is $\overline{\Delta V} > 0$, this constraint is written as:

$$|\Delta V_x(\nu_i)| \leqslant \overline{\Delta V}, \qquad |\Delta V_y(\nu_i)| \leqslant \overline{\Delta V}, \qquad |\Delta V_z(\nu_i)| \leqslant \overline{\Delta V}\ .$$

Space constraints on relative trajectories

During the rendezvous hovering phases, the follower spacecraft is required to steer and remain in the interior of a certain limited region of the space. The idea is to compute a sequence of N velocity corrections generating a relative trajectory that remains inside the hovering region during the interval $[t_N,\ t_{N+1}]$, as depicted in **Figure 7.3**.

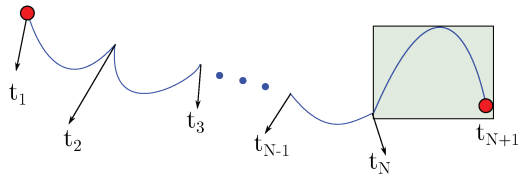


Figure 7.3: Steering into the hovering region within N velocity corrections

We assume in the sequel that this hovering range constraint is a rectangular cuboid:

$$x_{\min} \leqslant x(t) \leqslant x_{\max}, \quad y_{\min} \leqslant y(t) \leqslant y_{\max}, \quad z_{\min} \leqslant z(t) \leqslant z_{\max}, \quad \forall t \in [t_N, t_{N+1}]. \tag{7.7}$$

From (7.1), we have that:

$$\begin{bmatrix} x(t), & y(t), & z(t) \end{bmatrix} = \underbrace{\frac{1}{1 + e\cos\nu}}_{P(\nu)} \begin{bmatrix} \bar{x}(\nu), & \bar{y}(\nu), & \bar{z}(\nu) \end{bmatrix}.$$

Let us define $\mathbb{P}(\nu) > 0$ a positive RPA of $P(\nu)$ on the interval $[\nu_N, \nu_{N+1}]$ such that $|P^\circ(\nu) - P(\nu)| \leqslant \varepsilon_P,\ \forall \nu \in [\nu_N, \nu_{N+1}]$ and the RPAs $\mathbb{P}_x(\nu)$, $\mathbb{P}_y(\nu)$, $\mathbb{P}_z(\nu)$ defined as:

$$\begin{bmatrix} \mathbb{P}_x(\nu) & \mathbb{P}_y(\nu) & \mathbb{P}_z(\nu) \end{bmatrix}^T = \begin{bmatrix} \mathbf{1}_3 & \mathbf{0}_3 \end{bmatrix} \mathbb{X}(\nu).$$

Then, a RPA for the LVLH relative positions is given by:

$$\mathbb{x}(t) = \mathbb{P}_x(\nu) \boxslash \mathbb{P}(\nu), \quad \mathbb{y}(t) = \mathbb{P}_y(\nu) \boxslash \mathbb{P}(\nu), \quad \mathbb{z}(t) = \mathbb{P}_x(\nu) \boxslash \mathbb{P}(\nu).$$

The inequalities in (7.7) can be rigorously approximated by:

$$\begin{array}{ll} \mathbb{P}_x(\nu) - \mathbb{P}(\nu) x_{\min} \geqslant 0, & \mathbb{P}(\nu) x_{\max} - \mathbb{P}_x(\nu) \geqslant 0, \\ \mathbb{P}_y(\nu) - \mathbb{P}(\nu) y_{\min} \geqslant 0, & \mathbb{P}(\nu) y_{\max} - \mathbb{P}_y(\nu) \geqslant 0, \\ \mathbb{P}_z(\nu) - \mathbb{P}(\nu) z_{\min} \geqslant 0, & \mathbb{P}(\nu) z_{\max} - \mathbb{P}_z(\nu) \geqslant 0, \end{array} \quad \forall \nu \in [\nu_N, \nu_{N+1}].$$

Fuel-optimal impulsive control problem and MPC strategy

Using the mathematical models adopted for the constraints presented in the previous subsections, the fuel-optimal impulsive control problem that is iteratively solved in the model predictive control strategy is formulated as follows:

Problem 7.1 *Given $X(\nu_1) \in \mathbb{R}^6$, N true anomaly firing instants $\nu_1 < \ldots < \nu_N \in \mathbb{R}^+$, a true anomaly interval $[\nu_N, \nu_{N+1}]$ find $\Delta V^* \in [-\overline{\Delta V}, \overline{\Delta V}]^{3N}$ solution of:*

$$\begin{array}{ll} \underset{\Delta V}{\arg\min} & \mathcal{J}(\Delta V) \\ s.t. & \left\{ \begin{array}{ll} \mathbb{P}_x(\nu) - \mathbb{P}(\nu) x_{\min} \geqslant 0, & \mathbb{P}(\nu) x_{\max} - \mathbb{P}_x(\nu) \geqslant 0, \\ \mathbb{P}_y(\nu) - \mathbb{P}(\nu) y_{\min} \geqslant 0, & \mathbb{P}(\nu) y_{\max} - \mathbb{P}_y(\nu) \geqslant 0, \\ \mathbb{P}_z(\nu) - \mathbb{P}(\nu) z_{\min} \geqslant 0, & \mathbb{P}(\nu) z_{\max} - \mathbb{P}_z(\nu) \geqslant 0, \end{array} \right. \quad \forall \nu \in [\nu_N, \nu_{N+1}]. \end{array}$$

$(\mathcal{P}.\mathcal{SIP})$

This problem is a semi-infinite program (SIP), since the space constraints must be satisfied for infinitely many values of ν. However, the polynomial inequalities in $(\mathcal{P}.\mathcal{SIP})$ can be converted into so-called linear matrix inequalities (LMIs) using the results on the parametrization of non-negative polynomials on the cone of semi-definite positive matrices presented by **[182, Theorems 9 and 10]**, resulting in the semi-definite program (SDP) described in $(\mathcal{P}.\mathcal{SDP})$.

■ **Problem 7.2** *Given $X(\nu_1) \in \mathbb{R}^6$, N true anomaly firing instants $\nu_1 < ... < \nu_N \in \mathbb{R}^+$, a true anomaly interval $[\nu_N, \nu_{N+1}]$ find $\Delta V^* \in [-\overline{\Delta V}, \overline{\Delta V}]^{3N}$, Y_{1w}, $Y_{2w} \succeq 0$ solution of:*

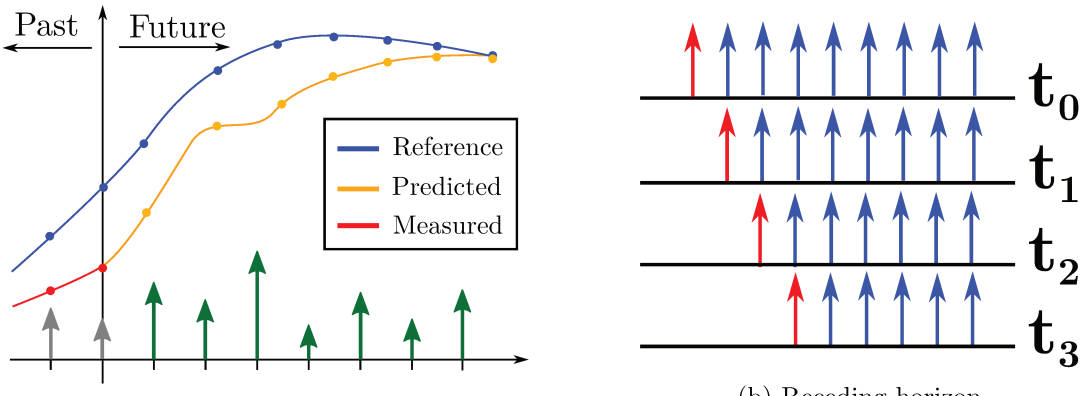
$$\begin{aligned} &\underset{\Delta V}{\arg\min} \quad \mathcal{J}(\Delta V) \\ &s.t. \quad \begin{cases} \widetilde{p}_{x_{\min}} = \Lambda^*(Y_{1x_{\min}}, Y_{2x_{\min}}), & \widetilde{p}_{x_{\max}} = \Lambda^*(Y_{1x_{\max}}, Y_{2x_{\max}}), \\ \widetilde{p}_{y_{\min}} = \Lambda^*(Y_{1y_{\min}}, Y_{2y_{\min}}), & \widetilde{p}_{y_{\max}} = \Lambda^*(Y_{1y_{\max}}, Y_{2y_{\max}}), \\ \widetilde{p}_{z_{\min}} = \Lambda^*(Y_{1z_{\min}}, Y_{2z_{\min}}), & \widetilde{p}_{z_{\max}} = \Lambda^*(Y_{1z_{\max}}, Y_{2z_{\max}}), \end{cases} \end{aligned} \qquad (\mathcal{P}.\mathcal{SDP})$$

*where $\widetilde{p}_w$ is the vector containing the coefficients of the respective non-negative polynomials in the SIP formulation and Λ^* is a bilinear operator (see more details in **[70, 69]**).*

The advantages of reformulating $(\mathcal{P}.\mathcal{SIP})$ are twofold:

- In the SIP formulation, the space constraints are described by infinitely many constraints on the true anomaly, requiring discretization techniques to efficiently compute a "solution". This solution, however, will systematically violate the constraints of the original problem. On the other hand, the SDP formulation provides a finite and exact description of the constraints;
- In previous works **[70, 7]**, SDP solvers were employed in the conception of control strategies for the spacecraft rendezvous problems, showing good performances even in environments with limited computational resources, such as devices dedicated to space applications.

The MPC strategy consists in iteratively solving and updating the inputs of $(\mathcal{P}.\mathcal{SDP})$ in a receding horizon manner. A sequence of N impulsive velocity corrections is computed each time the controller is called, but only the first impulse is applied. Then, the trajectory evolves freely until the next call of the controller (see **Figure 7.4**).



(a) Trajectory evolution and control actions

(b) Receding horizon

■ **Figure 7.4**: Illustration of model predictive control strategy and receding horizon.

Simulations and Results

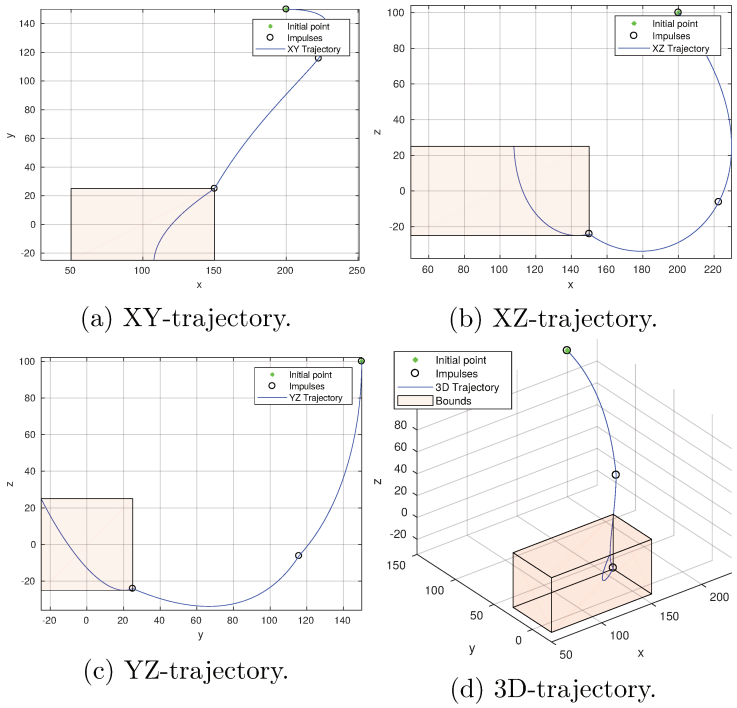
Hereafter we present the results obtained by solving one iteration of the MPC algorithm for the scenario described by the parameters given in **Table 7.1**.

Table 7.1: Scenario parameters

Semi-major axis:	$a = 7011$ km	Eccentricity:	$e = 0.4$
Initial true anomaly:	$\nu_1 = 0$ rad	Number of velocity corrections:	$N = 3$
Interval between impulses:	$\Delta\nu = \pi/4$ rad	Saturation threshold:	$\overline{\Delta V} = 1$ m/s
Initial relative state [m, m/s]:	$[200, 150, 100, 0, 0, 0]$	Degree of RPAs:	5, 7
$[x_{\min}, x_{\max}, y_{\min}, y_{\max}, z_{\min}, z_{\max}]$ [m]:	$[50, 150, -25, 25, -25, 25]$		

The RPAs are computed by the ChebValid C library, whereas the respective SDP problem is formulated on MATLAB via Yalmip[3] [160] and solved using the SDPT3 solver[4] [248]. The certificate enclosures are evaluated by treating the bounds of the RPAs using the b4m interval arithmetic toolbox library[5] [270].

In **Figure 7.5** the nominal relative trajectory obtained by simply propagating the initial state under the effect of the control actions is presented. **Figure 7.6**, **7.7** and **7.8** show the evolution of the x, z and y coordinates of the relative trajectory and their respective certificates.



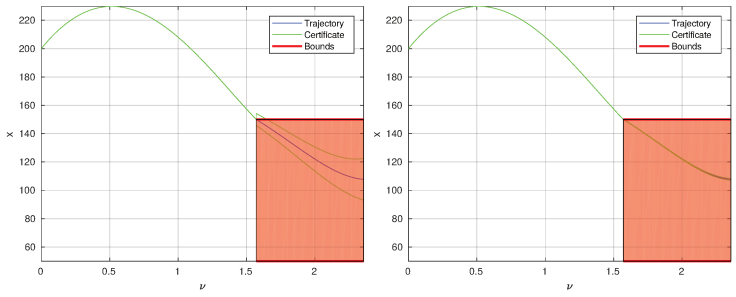
(a) XY-trajectory. (b) XZ-trajectory.

(c) YZ-trajectory. (d) 3D-trajectory.

Figure 7.5: Obtained relative trajectory without certification.

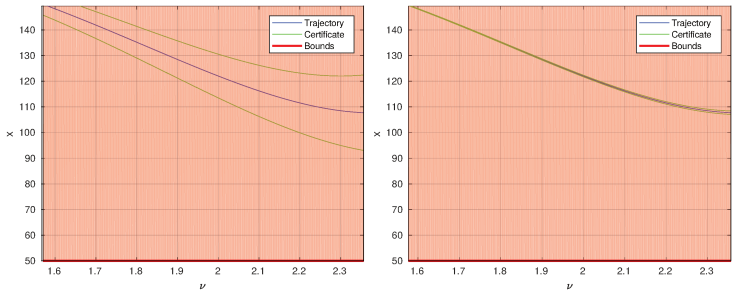
The **Figures** from **7.5a** to **7.5d** demonstrate that the computed impulses produce a relative trajectory that enters the hovering region and remains therein for the imposed true anomaly interval.

[3] `https://yalmip.github.io/`
[4] `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`
[5] `http://www.ti3.tu-harburg.de/zemke/b4m/`



(a) X-trajectory, RPAs of degree 5.

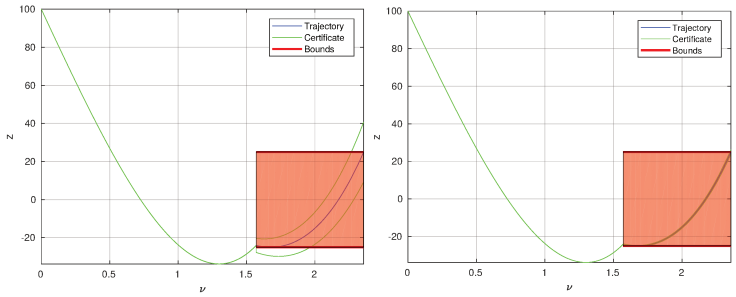
(b) X-trajectory, RPAs of degree 7.



(c) Final uncertainty for x-trajectory, RPAs of degree 5.

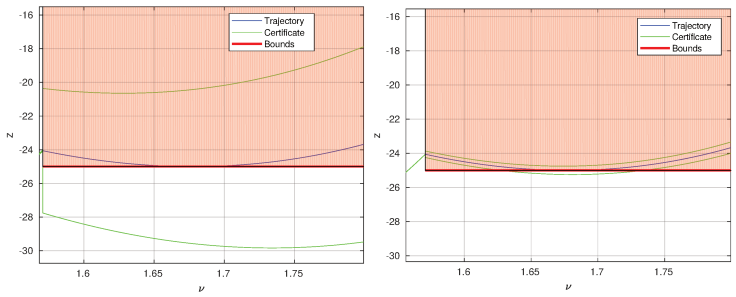
(d) Final uncertainty for x-trajectory, RPAs of degree 7.

■ **Figure 7.6**: X-coordinate in function of true anomaly for RPAs of degrees 5 and 7.



(a) Z-trajectory, RPAs of degree 5.

(b) Z-trajectory, RPAs of degree 7.

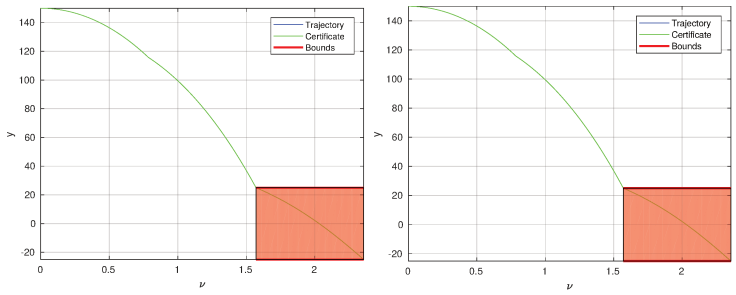


(c) Excursion of z-trajectory, RPAs of degree 5.

(d) Excursion of z-trajectory, RPAs of degree 7.

■ **Figure 7.7**: Z-coordinate in function of true anomaly for RPAs of degrees 5 and 7.

We can remark from the figures depicting the evolution of the in-plane x and z coordinates trajectories that the augmentation of the degree of the RPAs produce tighter certificate envelopes (in green) for the errors. This is well illustrated by comparing **Figure 7.7c** and **7.7d**: the RPAs of degree 5 produce a certificate of approximatively 5 m for the excursion of the trajectory, while the RPAs of degree 7 produce a much lower excursion certificate of approximatively 25 cm.



(a) Y-trajectory, RPAs of degree 5.

(b) Y-trajectory, RPAs of degree 7.

■ **Figure 7.8**: Y-coordinate in function of true anomaly for RPAs of degrees 5 and 7.

From **Figure 7.8a** and **7.8b** we observe that the certificates for y are tighter than those obtained for the in-plane coordinates x and y (the upper and lower certificates can not even be distinguished without zooming). This is explained by the fact that the out-of-plane dynamics is described by a simpler harmonic oscillator as previously presented in (7.5), which can be efficiently approximated by low degree polynomials. For instance, both RPAs produce final certificate enclosure of less than 2 cm for the relative trajectory.

7.3 Simple station keeping of a geostationary spacecraft

Let us now focus on obtaining a semi-analytical transition matrix, for a simple dynamics model, expressed in terms of the relative equinoctial orbital elements and linearized taking into account the J2 Earth oblateness effect.

The J2 perturbation is the C_{20} disturbing term of the decomposition of the Earth gravitational potential Legendre decomposition. This disturbing potential is expressed as:

$$\mathcal{E}_{C20} = \frac{\alpha_{20}}{r^3}\left(\frac{3}{2}\sin^2\varphi - \frac{1}{2}\right),$$

where r is the distance between the center of the Earth and the spacecraft, φ is the geographical latitude and $\alpha_{20} = \dfrac{\mu_\oplus r_\oplus^2 C_{20}}{2}$, $\mu_\oplus$ is the Earth gravitational parameter and $r_\oplus$ the mean Earth radius.

Following the derivation process described in the technical report [86], the disturbing potential is expressed in terms of the Cartesian position in the geocentric inertial reference frame, and then transformed into the equinoctial orbital elements. The differential equation of motion is derived using the Lagrange perturbation theory (see for instance the reference [16]), and then linearized. In Section 7.3.1 we give more details on these dynamics for the station keeping of a geostationary satellite with a low-thrust propulsion system, and the linearization we consider. In Section 7.3.2, we describe a new linearized model for the dynamics expressed with the relative equinoctial orbital elements. Then, in Section 7.3.3, we provide numerical examples for the validated polynomial transition matrices for the linearized dynamics occurring in this perturbed model.

7.3.1 ▶ Description of the model

The state vector of a satellite orbiting the Earth on a geostationary orbit is described with the equinoctial orbital elements as defined in [16]:

$$x_{eoe} = \begin{bmatrix} a & e_x & e_y & i_x & i_y & \ell_{M\Theta} \end{bmatrix}^T \in \mathbb{R}^6,$$

where a is the semi-major axis, (e_x, e_y) the eccentricity vector components, (i_x, i_y) the inclination vector components, $\ell_{M\Theta} = \omega + \Omega + M - \Theta$ is the mean longitude where Ω is the right ascension of the ascending node, ω is the perigee's argument, M is the mean anomaly and $\Theta(t)$ is the right ascension of the Greenwich meridian.

On top of the Keplerian gravitational attraction produced by the central body supposed to be spherical and homogeneous, spacecraft orbiting the Earth on a GEO orbit undergo orbital disturbing forces. In [230], [233], [231], the potential function of these orbital perturbations is expressed by means of the geographical positions, i.e. radius, latitude and longitude of the spacecraft and the disturbing bodies. As the chosen state vector for the GEO spacecraft is composed of the equinoctial orbital elements, it is mandatory to transform the expression of these potential function in terms of variables of the state vector. In the technical report [86] the geographical position is first transformed in the Cartesian position in the geocentric inertial reference frame referred as the ECI reference frame in [249], and then transformed in the equinoctial orbital elements thanks to conversion formulas derived in the Appendix C of [86].

The disturbing effects described before make the satellite drift away from its nominal position. It is therefore mandatory to equip the satellite with thrusters in order to correct the satellite orbit. These thrusters also create a disturbing acceleration.

With the equinoctial orbital elements as state variables, the dynamic equation to handle the orbital perturbation is given by the Lagrange perturbation technique and the dynamic equation for the effect of the thrusters by the Gauss variation technique (see for instance the references [268] or [161]). By superposition principle, the two effects can be added, leading to the following dynamic equation:

$$\frac{\mathrm{d}x_{eoe}}{\mathrm{d}t} = f_L(x_{eoe}, t) + f_G(x_{eoe}, t)u. \tag{7.8}$$

where $f_L \in \mathbb{R}^6$ is the Lagrange contribution part of the external forces and $f_G \in \mathbb{R}^{6\times 3}$ is the

Gauss contribution part. $u = [u_R \ u_T \ u_N]^t \in \mathbb{R}^3$ is the control vector expressed in the local orbital frame.

7.3.2 ▶ Linearization

For operational purposes, the satellite has to stay on an operating position called station keeping point. As the orbital disturbances induce a drift of the spacecraft position, the thrusters are fired in order to make the spacecraft stay in the vicinity of its operating position in a so-called station keeping window, whose size is very small with respect to the distance to the Earth. It is therefore possible to linearize the nonlinear Equation (7.8) with respect to the orbital elements of the station keeping point:

$$x_{sk} = [a_{sk} \ 0 \ 0 \ 0 \ 0 \ \ell_{M\Theta_{sk}}]^T, \tag{7.9}$$

where a_{sk} is the synchronous semi-major axis and $\ell_{M\Theta_{sk}}$ is the station mean longitude. This station keeping state is a fictitious point evolving on a Keplerian (unperturbed) GEO orbit. It is defined such that the spacecraft mean motion equals the Earth rotation rate. It is then straightforward that:

$$\frac{\mathrm{d}x_{sk}}{\mathrm{d}t} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \sqrt{\dfrac{\mu}{a_{sk}^3}} - \omega_T \end{bmatrix}^T = 0,$$

where ω_T is the Earth rotation rate. Therefore, $a_{sk} = 42164$ km. Moreover, for the simulations, we choose $\ell_{M\Theta_{sk}} = 118°$.

The relative dynamics equations are derived by a new linearization of Equation (7.8) about the station keeping point (7.9). Denoting $x = x_{eoe} - x_{sk}$, the relative state model for the SK problem is computed as follows:

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} = \frac{\mathrm{d}x_{eoe}}{\mathrm{d}t} - \frac{\mathrm{d}x_{sk}}{\mathrm{d}t} &= f_L(x_{eoe}, t) + f_G(x_{eoe}, t)u - 0 \\ &\approx f_L(x_{sk}, t) + \left.\frac{\partial f_L(x_{eoe}, t)}{\partial x_{eoe}}\right|_{x_{eoe}=x_{sk}} x + f_G(x_{sk}, t)u. \end{aligned} \tag{7.10}$$

From Equation (7.10) the dynamical model reads:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = A(t)x + D(t) + B(t)u, \tag{7.11}$$

where the matrices $A \in \mathbb{R}^{6\times6}$, $B \in \mathbb{R}^{6\times3}$ and $D \in \mathbb{R}^6$ are defined as follows:

$$A(t) = \left.\frac{\partial\left(f_L(x_{eoe}(t), t)\right)}{\partial x_{eoe}}\right|_{x_{eoe}=x_{sk}}, \qquad B(t) = f_G(x_{sk}, t), \qquad D(t) = f_L(x_{sk}, t).$$

As the dynamics of the relative equinoctial elements given by the Equation (7.11) is now linear, the effects of each perturbation can be added together, such that:

$$\begin{aligned} A(t) &= A_{Keplerian}(t) + A_{J2}(t), \\ D(t) &= D_{Keplerian}(t) + D_{J2}(t), \end{aligned} \tag{7.12}$$

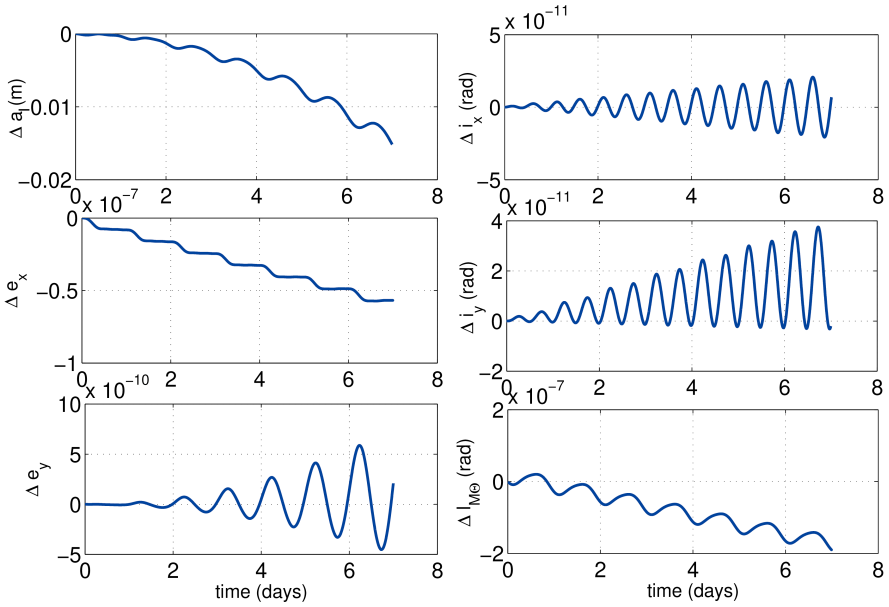
with the exact expression of these matrices available in [86]. From Equations (7.12) and (7.11), the (uncontrolled) state transition matrix is computed with our method. In what follows, a numerical integration example is given to show its effectiveness. Note that this state transition matrix can then be used in the framework of the linear GEO station keeping control problem, which is expressed as an optimal control problem whose objective is to minimize the fuel consumption while ensuring that the (linearized) relative dynamic is respected and that the spacecraft does not fly out its station keeping window [161], [88, 90] and [87].

7.3.3 ▶ Numerical example of integration

In this example, the integration has been performed for an initial relative equinoctial orbital elements state:

$$X_0 = \begin{bmatrix} 0 & 10^{-4} & 0 & 10^{-4} & 0 & 0 \end{bmatrix}^T.$$

In **Figure 7.9** the difference between the relative equinoctial elements integrated with the non-linear equation of motion and the linearized equation of motion with the `ode45` function of MATLAB is presented on a time interval $[t_0, t_f]$ with $t_0 = 0$ and $t_f = 7$ days. One observes that for the semi-major axis, the two trajectories diverge one from the other up to 0.015 m after seven days. This is due to the fact that the spacecraft moves away from the station keeping point. As for station keeping purposes, a control law will enforce the spacecraft to stay in the vicinity of the station keeping point, the approximation error between the linear and the nonlinear model should remain small. The error between the linear and nonlinear integration after 7 days is representative of the error that will occur during the station keeping control process because the uncontrolled trajectory flies out the station keeping window after 7 days of free motion.



■ **Figure 7.9**: Error between the integration of the relative equinoctial orbital elements with the nonlinear and the linearized model.

Based on this, the magnitude of the maximum error obtained between the integrated non-linear equation of motion and the linearized one are heuristically estimated and used as benchmarks in **Table 7.2** in order to compute the minimal degree of the rigorous polynomial approximations (obtained with our method) which achieve them. Specifically, associated with equation (7.11), consider the solution $x(t) = \Phi(t,t_0)x_0 + \int_{t_0}^{t} \Phi(t,s)D(s)\mathrm{d}s$, for $t \in [t_0, t_f]$, where $\Phi(t,t_0)$ is the transition matrix and $\int_{t_0}^{t} \Phi(t,s)D(s)\mathrm{d}s$ is a particular solution. The first 6 columns in **Table 7.2** deal with RPAs approximating the transition matrix $\Phi(t,t_0)$. For instance, for the first entry of the transition matrix, a maximum error upper-bound of 10^{-10} requires a degree-26 RPA. Similarly, the last column of **Table 7.2** shows respectively the maximum error upper-bounds and the required degree for RPAs for the particular solution.

	j						
i	1	2	3	4	5	6	$\star$
1	10^{-10} : **26**	10^{-7} : **30**	10^{-10} : **37**	10^{-10} : **0**	10^{-10} : **0**	10^{-7} : **0**	10^{-11} : **28**
2	10^{-7} : **33**	10^{-7} : **52**	10^{-7} : **52**	10^{-7} : **0**	10^{-7} : **0**	10^{-7} : **29**	10^{-11} : **51**
3	10^{-10} : **48**	10^{-7} : **52**	10^{-10} : **61**	10^{-10} : **0**	10^{-10} : **0**	10^{-7} : **29**	10^{-11} : **51**
4	10^{-10} : **0**	10^{-7} : **0**	10^{-10} : **0**	10^{-11} : **63**	10^{-11} : **64**	10^{-7} : **0**	10^{-11} : **0**
5	10^{-10} : **0**	10^{-7} : **0**	10^{-10} : **0**	10^{-11} : **64**	10^{-11} : **63**	10^{-7} : **0**	10^{-11} : **0**
6	10^{-10} : **0**	10^{-7} : **0**	10^{-10} : **0**	10^{-11} : **64**	10^{-11} : **63**	10^{-7} : **0**	10^{-11} : **0**

Table 7.2: Maximum error bounds required for each entry of the transition matrix (columns $j = 1, \ldots, 6$) and particular solution (column $j = \star$), and minimal degrees of the rigorous polynomial approximations achieving them, obtained with our method. **Remark:** a null degree indicates a constant entry.

Figure 7.10 depicts the difference between the relative equinoctial orbital elements computed on one hand by integration of the linearized dynamic given by Equation (7.11) with the `ode45` function of MATLAB and on the other hand by the semi-analytical state transition matrix computed by RPAs. The i_y and $\ell_{M\Theta}$ components undergo a secular error whereas the other ones present small periodic errors. Nevertheless, the relative error between these two trajectories is smaller than the error between the linear and the non-linear integration of the trajectory, justifying the use of these state transition matrices as a way to compute the relative trajectory.

7.4 Conclusion and future developments

A validated MPC algorithm for the rendezvous hovering phases has been conceived using the proposed approximation method. Future experiments would assess the tractability of problem ($\mathcal{P}.\mathcal{SDP}$) on devices dedicated to space applications, focusing on the analysis of the relation between the computational burden and the precision of the polynomial approximations. The study of the performances of problems similar to **Section 7.2.2** has already been carried out

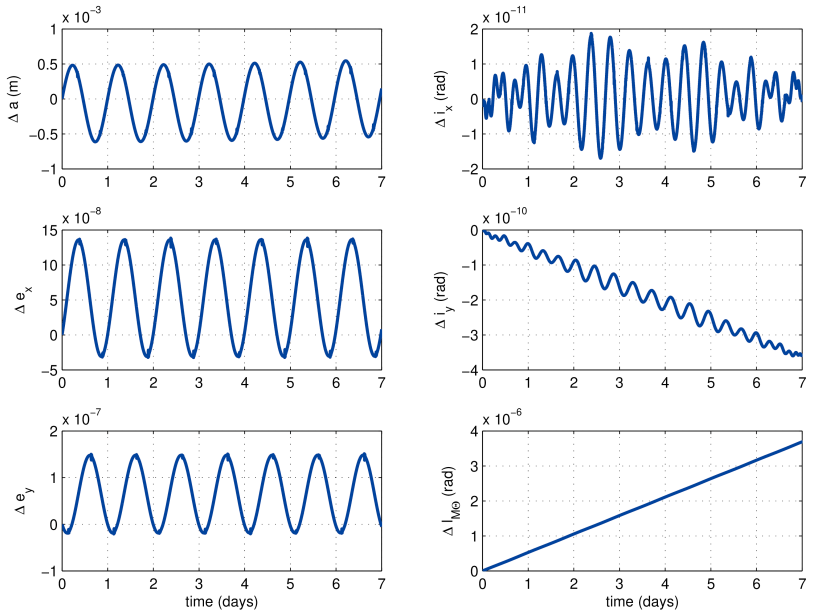


Figure 7.10: Error between the integration of the relative equinoctial orbital elements with the `ode45` function of MATLAB and the proposed semi-analytical state transition matrix computation method.

using an AEROFLEX GAISLER GR-XC6S board containing a synthesized LEON3 microprocessor in [70, 7].

The proposed approximation technique has then been applied on a more complicated case where orbital disturbances arise. Although only the most prominent perturbation has been handled, the proposed state transition matrices computation method could also been applied to other orbital disturbances, as for instance the Sun and Moon gravitation attractions or the Sun radiation pressure. These effects leave room for improvement to our method because it would be necessary to take into account the Sun and Moon positions that are known as tabulated functions. The references [110] or [25] describe how direct collocation methods can be used in order to solve the GEO station keeping optimal control problem. These methods rely on a discretization of the state and control vectors over the time interval $[t_0, t_f]$. The optimal control problem is therefore transformed into a nonlinear programming problem. The dimension of the unknown vector for the nonlinear programming problem can be reduced while eliminating the state vector, meaning that the state differential equation of the system must be integrated explicitly. The proposed technique for the computation of the state transition matrix will therefore be used for the integration of the dynamic equation, making the resolution of the GEO station keeping optimal control problem easier.

Another possible extension for this work would be the propagation of uncertain initial conditions via semi-analytical polynomial transition matrices. When the uncertainties in the initial conditions are not uniformly distributed, we plan consider the generalization to other classes of orthogonal polynomials.

Exchange Algorithm for Evaluation & Approximation Error-Optimized Polynomials

8

Comment voulez-vous gouverner un pays où il existe 258 variétés de fromage ?
— Charles de Gaulle

Machine implementation of mathematical functions often relies on polynomial approximations, discussed in **Chapter 2**. The particularity is that rounding errors occur both when representing the polynomial coefficients on a finite number of bits, and when evaluating it in finite precision. Hence, for finding the best polynomial (for a given fixed degree, norm and interval), one has to consider both types of errors: approximation and evaluation. While efficient algorithms were already developed for taking into account the approximation error, the evaluation part is usually a posteriori handled, in an ad-hoc manner.

In this chapter, which is a joint work with Denis Arzelier and Mioara Joldes, we formulate a semi-infinite linear optimization problem whose solution is the best polynomial with respect to the supremum norm of the sum of both errors. This problem is then solved with an iterative exchange algorithm, which can be seen as an extension of Remez algorithm, recalled in **Section 2.2.2**. A discussion and comparison of the obtained results on various examples are finally presented.

This work gave rise to an article entitled "Exchange algorithm for evaluation and approximation error-optimized polynomials" **[13]**, to be published in the proceedings of the *26th IEEE Symposium on Computer Arithmetic* (ARITH).

8.1 General setting and contributions

Polynomials are often used for approximating functions on computers [2, 176]. Their evaluation only requires additions and multiplications, which are efficiently implemented in hardware floating-point (FP) arithmetic units.

As mentioned in Chapter 1, FP operations are specified by the IEEE 754-2008 [113] standard, which requires, among others, correctly rounded basic arithmetic operations $+, -, *, /, \sqrt{}$ for several precision formats, and recommends correctly rounded elementary functions like exp, sin, cos. Very efficient fixed FP precision implementations exist [68, 151] for such functions and are collected in mathematical libraries (*libm*s), which can be nowadays almost automatically generated and tuned [64, 144]. Recently, in [152], such code generating techniques were extended to larger classes of special functions, which are widely used in scientific and technical applications (like Bessel, Airy, Erf, etc.).

The problem of evaluating a function for the whole FP input range is firstly reduced to the evaluation of an approximation valid in a rather small compact domain I. This can be done for instance, by argument reduction techniques, which are available only for specific elementary functions, and/or by piecewise polynomial approximations [177]. Then, the implementation task becomes: given a description of a function f, an input interval I, and a target accuracy $\varepsilon > 0$, one is requested a source code which provides a function $\tilde{f}$, such that: $\left\|(f-\tilde{f})/f\right\|_I \leqslant \varepsilon$, where we denote by $\|g\|_I := \sup_{t\in I} |g(t)|$ the supremum norm of g on I.

Typically, this is handled in two main steps:

Approximation. An approximation polynomial p is searched for, such that two main requirements are met: its coefficients are representable with a specified fixed precision format (usually, binary32, binary64, or an unevaluated sum of such formats) and the approximation error is less than a target $\varepsilon_{\text{approx}}$, whether absolute $\|f-p\|_I \leqslant \varepsilon_{\text{approx}}$ or relative $\|(f-p)/f\|_I \leqslant \varepsilon_{\text{approx}}$.

For that, efficient algorithms were developed, e.g., [42] for low degree and [43] for larger degrees. In the simpler case of polynomials p with real coefficients and given degree n, $p = \sum_{i=0}^{n} a_i t^i$, this boils down to the so-called *minimax* problem, addressed in Chapter 2:

$$\min_{\substack{a_i\in\mathbb{R},\\ i\in[\![0,n]\!]}} \max_{t\in I} |f(t)-p(t)|, \qquad (P_{\text{minimax}})$$

which can be solved by the Remez algorithm (see Section 2.2.2, [43, 54] and references therein). This iterative algorithm has quadratic convergence and rather low complexity, since it involves solving a linear system of size $n+2$ at each step, together with numerically computing the extrema of $f-p$ over I.

Evaluation. An efficient evaluation scheme $\tilde{p}$ for p is searched for; since after each addition or multiplication, rounding errors occur, one must ensure that the computed value satisfies $\|p-\tilde{p}\|_I \leqslant \varepsilon_{\text{eval}}$ (or $\|(p-\tilde{p})/p\|_I \leqslant \varepsilon_{\text{eval}}$) for a given threshold $\varepsilon_{\text{eval}}$.

Heuristics presented in [151] extend the precision of the *important* coefficients, such that the evaluation error remains below $\varepsilon_{\text{eval}}$. For instance, Sollya command `implementpoly` uses

a Horner-based evaluation scheme, which behaves rather well when the evaluation interval is sufficiently small and contains zero. Otherwise, consider step i of Horner evaluation $a_i + t\tilde{p}_i(t)$, where $\tilde{p}_i$ is the already computed partial polynomial evaluation: when the argument $|t| \gg 1$, the accumulated evaluation error is much amplified when multiplying by t. Another heuristic is a ratio test between a_i and $t\tilde{p}_i(t)$, to check for cancellation issues which appear when both terms have the same order of magnitude and opposite signs.

Once the coefficients have been chosen, the approximation and the evaluation error can *a posteriori* be certified by several existing algorithms and tools, like Sollya [55], Gappa [66], Rosa [65] or Real2Float [215].

It is important to note that these two steps are usually independently considered. An exception occurs for the case of very small precisions or polynomial degrees, where an exhaustive search on the rounded coefficients is possible [242].

However, as explicitly mentioned in [42], "*one would like to take into account the roundoff error that occurs during polynomial evaluation: getting the polynomial, with constraints on the size of the coefficients, that minimizes the total (approximation plus roundoff) error would be extremely useful*".

The purpose of this article is to make progress on this open question: we search for the coefficients of a polynomial $p(t) = \sum_{i=0}^{n} a_i t^i$, of given degree n, which minimizes the maximum of the sum of both approximation and evaluation errors over an input interval I, with respect to f. We consider a *black-box* description of f i.e., one can ask for the value $f(t)$ for any desired t, up to any required accuracy [151]. This allows for handling very general functions (elementary, special, etc.), but also implies that no argument reduction step is possible in general. For simplicity we state the problem for the absolute error case (The relative error can be similarly handled from a theoretical standpoint, cf. **Section 8.4.3**):

$$\min_{\substack{a_i \in \mathbb{R}, \\ i \in [\![0,n]\!]}} \max_{t \in I} \left(|f(t) - p(t)| + |\tilde{p}(t) - p(t)| \right) \qquad (P_{\text{general}})$$

Remark 8.1 (Floating-point coefficients) *The formulation of* ***Problem*** (P_{general}) *seems to ignore the constraints that the coefficients of the approximating polynomial must be floating-point representable. In fact, even if we optimize over real coefficients, the effect of rounding them can be incorporated in the evaluation error term* $|\tilde{p}(t) - p(t)|$*. Hence, the objective function of this optimization problem takes into account the constraint of floating-point coefficients.*

In **Section 8.2**, we give a linearized bound for the evaluation error $|\tilde{p}(t) - p(t)|$. Based on [190], the performance of a given arbitrary evaluation scheme is recursively assessed by bounding the rounding error of each elementary operation. This leads to the formulation in **Section 8.3** of **Problem** (P_{general}) as a linear semi-infinite programming (SIP) problem [207, 228].

In this context, we show two results: on the theoretical side, based on the duality theory, we revisit, explain and extend an *exchange algorithm* [259, 50, 48, 49], which solves this problem in **Section 8.4**. On the practical side, the solution of this problem provides a first attempt on simultaneously optimizing over both errors: we show that in some cases the evaluation error can be improved. We also show that in some other cases, the minimax polynomial solution of **Problem** (P_{minimax}) is very close to the solution of P_{general}. Numerical examples and a discussion are provided in **Section 8.5**.

8.2 | Evaluation error

Throughout this chapter, we make use of the definitions and notations from **Chapter 1** about floating-point arithmetic. We assume radix-2, precision-p, floating-point arithmetic with unbounded exponent range i.e, provided that overflows and underflows do not occur. Recall in particular that for any $t \in \mathbb{R}$, we have

$$\frac{|t - \mathbf{RN}(t)|}{|t|} \leqslant \frac{u}{1+u} < u,$$

where $u = 2^{-p}$ is called the *rounding unit*. Moreover, for each $\top \in \{+, -, \times, /, \sqrt{}\}$, there exists a real number ϵ such that

$$\mathbf{RN}(a \top b) = (a \top b)(1+\epsilon), \quad |\epsilon| \leqslant u.$$

Based on the previous property, the error of any arithmetic expression can be recursively bounded. This leads to computable evaluation errors associated to any evaluation scheme $\tilde{p}$ for a polynomial p. Firstly, for specific evaluation schemes, like Horner, bounds date back to the work of Oliver **[190]**, which is detailed below as an example. More recently, several works insisted on the automatic algorithmic approach via operator overloading similar to automatic differentiation **[36, 232]**. Based on this, we propose, for completeness[1], **Algorithm** LinEvalError, which automatically computes linearized expressions for the evaluation error (like in **(8.3)**), for any given symbolic expression tree e, provided with symbolic rounding errors for each tree node. Let us exemplify on the evaluation of the polynomial $p(t) = a_n t^n + a_{n-1}t^{n-1} + \cdots + a_0$

Algorithm 8.1 Horner(p, t) – Classical Horner scheme

1: $r_n \leftarrow a_n$
2: **for** $k = n-1$ **downto** 0 **do**
3: $\quad r_k \leftarrow \mathbf{RN}\big(\mathbf{RN}(r_{k+1} \times t) + a_k\big)$
4: **end for**
5: **return** r_0

using Horner's rule, assuming that a Fused Multiply Add (FMA) instruction is not employed. The actual machine operations are recalled in **Algorithm** Horner. We have:

$$\begin{aligned} r_n &= a_n, \\ r_{n-1} &= \big(t r_n (1+\epsilon_{n-1}^{\times}) + a_{n-1}\big)(1+\epsilon_{n-1}^{+}), \end{aligned}$$

where $\epsilon_{n-1}^{\times}$ and ϵ_{n-1}^{+} model the rounding errors for multiplication and addition at step $n-1$. By induction, one obtains:

$$r_k = \sum_{i=k}^{n} \left((1+\epsilon_i^{+}) \prod_{j=k}^{i-1} (1+\epsilon_j^{+})(1+\epsilon_j^{\times}) \right) a_i t^{i-k}, \tag{8.2}$$

[1] While the general ideas are the same as in **[36, 232]** and references therein, we could not find the exact pseudo-code in literature, so it is stated in order to provide a complete algorithmic solution for **Problem** (P_{general}).

where we define $\epsilon_n^+ := 0$ and $\prod_{j=k}^{k-1}(1+\epsilon_j^+)(1+\epsilon_j^\times) := 1$. This implies that the total evaluation error is:

$$r_0 - \sum_{i=0}^{n} a_i t^i = \sum_{i=0}^{n} \left((1+\epsilon_i^+) \prod_{j=k}^{i-1} (1+\epsilon_j^+)(1+\epsilon_j^\times) - 1 \right) a_i t^i.$$

Here, we consider only a linear approximation θ_{lin} of the evaluation error, function of ϵ_i^+ and $\epsilon_i^\times$, in what follows. This gives, for our Horner example:

$$\theta_{\text{lin}}^{(\text{Horner})} := \sum_{j=0}^{n-1} \left(\sum_{i=j+1}^{n} a_i t^i \right) \epsilon_j^\times + \sum_{j=0}^{n-1} \left(\sum_{i=j}^{n} a_i t^i \right) \epsilon_j^+. \tag{8.3}$$

Moreover, provided bounds are specified for each rounding error, depending on the precision employed, one obtains upper bounds for the linearized absolute evaluation error. For instance, if binary64 is used for all the computations in **Algorithm** HORNER, with $u = 2^{-53}$, one has:

$$\left| \theta_{\text{lin}}^{(\text{Horner})} \right| \leqslant 2u \sum_{j=0}^{n}{}'' \left| \sum_{i=j}^{n} a_i t^i \right|, \tag{8.4}$$

where the double superscript indicates that the first and last terms in the summation are to be halved.

As exemplified in **Table 8.1**, to automate the evaluation error analysis, we firstly associate to a mathematical expression e^*, a given symbolic evaluation scheme with roundings e, composed of terms $\mathbf{RN}(e', u)$. This means that e' is rounded with a relative error bounded by u. This formulation models both possible rounding errors on an input variable ($e' \in \mathcal{V}$, where $\mathcal{V}$ denotes the set of input variables) and the rounding errors of arithmetic operations (if $e' = a_1 \top e_2$). Then, we build an expression $\tilde{e}$, as in (8.2), by recursively replacing terms $\mathbf{RN}(e', u)$ in e, with $\tilde{e}'(1 + \epsilon_{e'}^{[u]})$ where $|\epsilon_{e'}^{[u]}| \leqslant u$.

$e_1 = \mathbf{RN}(a + \mathbf{RN}(b \times c, u), u)$	$e_1^* = a + bc$
$\tilde{e}_1 = (a + bc(1 + \epsilon_{b\times c}^{[u]}))(1 + \epsilon_{a+\mathbf{RN}(b\times c,u)}^{[u]})$	
$\theta_{\text{lin}}^{(e_1)} = bc\epsilon_{b\times c}^{[u]} + (a + bc)\epsilon_{a+\mathbf{RN}(b\times c,u)}^{[u]}$	
$\lvert\theta_{\text{lin}}^{(e_1)}\rvert \leqslant (\lvert bc\rvert + \lvert a + bc\rvert)u$	
▷ *Arithmetic operations in double precision.*	
$e_2 = \mathbf{RN}(a + b \times c, u)$	$e_2^* = a + bc$
$\tilde{e}_2 = (a + bc)(1 + \epsilon_{a+b\times c}^{[u]})$	
$\theta_{\text{lin}}^{(e_2)} = (a + bc)\epsilon_{a+b\times c}^{[u]}$	
$\lvert\theta_{\text{lin}}^{(e_2)}\rvert \leqslant \lvert a + bc\rvert u$	
▷ *Fused multiply-add (FMA) in double precision.*	
$e_2 = \mathbf{RN}(\mathbf{RN}(a, u') + b \times \mathbf{RN}(a, u'), u)$	$e_2^* = a + ba$
$\tilde{e}_2 = (a(1 + \epsilon_a^{[u']}) + ba(1 + \epsilon_a^{[u']}))(1 + \epsilon_{\mathbf{RN}(a,u')+b\times\mathbf{RN}(a,u')}^{[u]})$	
$\theta_{\text{lin}}^{(e_2)} = (a + ba)\epsilon_a^{[u']} + (a + ba)\epsilon_{\mathbf{RN}(a,u')+b\times\mathbf{RN}(a,u')}^{[u]}$	
$\lvert\theta_{\text{lin}}^{(e_2)}\rvert \leqslant \lvert a + ba\rvert(u + u')$	
▷ *Fused multiply-add (FMA) in double precision, with input a rounded to single precision.*	

■ **Table 8.1**: Evaluation error examples ($u = 2^{-53}$, $u' = 2^{-24}$).

Finally, automatic linearized evaluation error expressions θ_{lin}, such as in (8.3), are obtained using **Algorithm** LinEvalError. Specifically, for an arithmetic expression with roundings e, this algorithm recursively computes an expression of the form $\theta_{\text{lin}} = \sum_{i=1}^{k} \theta_{\text{lin},i}\epsilon_{e_i}^{[u_i]}$, with symbolic $\epsilon_{e_i}^{[u_i]}$ ($i \in [\![1, k]\!]$) for each term $\mathbf{RN}(e_i, u_i)$ in e. The coefficients $\theta_{\text{lin},i}$ are arithmetic expressions depending only on the input variables in e. Note that $\mathbf{RN}(e_i, u_i)$ may occur several times in e, but the error variable $\epsilon_{e_i}^{[u_i]}$ is unique since the rounding operation is deterministic. This allows us to bound the (linearized) evaluation error as in (8.4).

■ **Proposition 8.2** (Correctness of **Algorithm** LinEvalError) *Let e be an arithmetic expression with roundings, and $\theta_{lin} = \sum_{i=1}^{k} \theta_{lin,i}\epsilon_{e_i}^{[u_i]}$ the linearized expression for the evaluation error returned by **Algorithm** LinEvalError. If $u_i \leqslant u$ for all $i \in [\![1, k]\!]$, then:*

$$|\tilde{e} - e^*| \leqslant \sum_{i=1}^{k} |\theta_{lin,i}|u_i + \mathcal{O}\left(u^2\right), \qquad as\ u \to 0.$$

Usually, for polynomial evaluation schemes, the functions $\theta_{\text{lin},i}$ are linear with respect to the coefficients $\boldsymbol{a}$ of $p(t)$, that is $u_i\theta_{\text{lin},i} = \boldsymbol{\pi}_i(t)^T\boldsymbol{a}$, with $\boldsymbol{a} \in \mathbb{R}^{n+1}$, $t \in \mathbb{R}$ and for some $\boldsymbol{\pi}_i(t) \in \mathbb{R}^{n+1}$. Hence we obtain a linearized bound of the evaluation error of the form:

$$|\theta_{\text{lin}}(\boldsymbol{a}, t)| \leqslant \sum_{i=1}^{k} |\boldsymbol{\pi}_i(t)^T\boldsymbol{a}| := \theta(\boldsymbol{a}, t). \tag{8.5}$$

Example 8.3 *For Horner evaluation, Equation (8.4) gives:*

$$\begin{aligned}
\boldsymbol{\pi}_1(t)^T &= (u, ut, \ldots, ut^{n-1}, ut^n),\\
\boldsymbol{\pi}_2(t)^T &= (0, 2ut, \ldots, 2ut^{n-1}, 2ut^n), \ldots,\\
\boldsymbol{\pi}_n(t)^T &= (0, 0, \ldots, 2ut^{n-1}, 2ut^n),\\
\boldsymbol{\pi}_{n+1}(t)^T &= (0, 0, \ldots, 0, ut^n).
\end{aligned}$$

Remark 8.4 *The use of **Algorithm** LinEvalError is flexible and allows for the treatment of various situations, e.g., evaluation schemes with several floating-point precisions or compensated schemes. The user just needs to specify the accuracy of each floating-point operation.*

Algorithm 8.2 LinEvalError(e) – Linearized absolute rounding error

Input: e an arithmetic expression with explicit roundings.
Output: θ_{lin} the linearized evaluation error of e.

if $e \in \mathcal{V}$ **then**
  return 0
else if $e = \mathsf{RN}(f, u)$ **then**
  $\theta'_{\text{lin}} \leftarrow$ LinEvalError(f)
  return $\theta'_{\text{lin}} + f^* \epsilon_f^{[u]}$
else if $e = -f$ **then**
  $\theta'_{\text{lin}} \leftarrow$ LinEvalError(f)
  return $-\theta'_{\text{lin}}$
else if $e = f + g$ **then**
  $\theta'_{\text{lin}} \leftarrow$ LinEvalError(f)
  $\theta''_{\text{lin}} \leftarrow$ LinEvalError(g)
  return $\theta'_{\text{lin}} + \theta''_{\text{lin}}$
else if $e = f \times g$ **then**
  $\theta'_{\text{lin}} \leftarrow$ LinEvalError(f)
  $\theta''_{\text{lin}} \leftarrow$ LinEvalError(g)
  return $g^* \theta'_{\text{lin}} + f^* \theta''_{\text{lin}}$
end if

8.3 Semi-Infinite Programming formulation

Problem (P_{general}) is rephrased in the framework of semi-infinite programming (SIP) in **Section 8.3.1**. Then, some duality and discretization properties are summarized in **Section 8.3.2**, leading to the exchange algorithm presented in **Section 8.4**.

8.3.1 ▶ Formulation as a linear SIP

Noting that **Problem** (P_{general}) is a piecewise-linear optimization problem and using the convex evaluation error formula $\theta(\boldsymbol{a}, t)$ at point $t \in [t_l, t_r]$ obtained in **Section 8.2**, **Problem** (P_{general}) becomes **Problem** (P'_{general}) (see **[35,** Section 4.3.1**]** for instance), with the compact index set $I = [t_l, t_r]$ and the monomial basis $\boldsymbol{\pi_0}(t) = (1, \dots, t^n)^T$.

$$\begin{array}{ll} \min\limits_{(\overline{a},\boldsymbol{a})\in\mathbb{R}^{n+2}} & \overline{a} \\ \text{s.t.} & |f(t) - \boldsymbol{\pi_0}(t)^T\boldsymbol{a}| + \theta(\boldsymbol{a}, t) - \overline{a} \leqslant 0, \quad t \in I. \end{array} \tag{P'_{general}}$$

Problem (P'_{general}) is a convex Semi-Infinite Programming (SIP) problem (see **[207]** which provides a comprehensive overview of SIP) that can be reformulated as a linear SIP problem, at the expense of a different index set Ω replacing the previous index set I. Here, the set of constraints of (P'_{general}) involving absolute values is replaced by as many linear constraints as required to represent all possible sign combinations. The evaluation error is as in Equation (8.5), and define:

$$\boldsymbol{x} := (\overline{a}, \boldsymbol{a}) \in \mathbb{R}^{n+2}, \qquad \boldsymbol{z} := (1, 0, \dots, 0) \in \mathbb{R}^{n+2},$$

$$\boldsymbol{\alpha}(t, \sigma_0, \dots, \sigma_k) := (1, \sigma_0 \boldsymbol{\pi_0}^T(t) + \sum_{i=1}^{k} \sigma_i \boldsymbol{\pi_i}^T(t))^T \in \mathbb{R}^{n+2},$$

$$\mathfrak{S} := \{-1, 0, 1\}^{k+1}, \qquad \omega := (t, \sigma_0, \dots, \sigma_k) \in \Omega := I \times \mathfrak{S}.$$

Then, **Problem** (P'_{general}) is exactly the following linear SIP:

$$\begin{array}{ll} \min\limits_{\boldsymbol{x}\in\mathbb{R}^d} & \boldsymbol{z}^T\boldsymbol{x} \\ \text{s.t.} & \boldsymbol{\alpha}(\omega)^T\boldsymbol{x} \geqslant c(\omega), \quad \omega \in \Omega, \end{array} \tag{P}$$

where $d = n + 2$, $c(\omega) = \sigma_0 f(t)$, Ω is a compact metric space and the function $g(\boldsymbol{x}, \omega) = c(\omega) - \boldsymbol{\alpha}(\omega)^T\boldsymbol{x} \leqslant 0$ defining the feasible set is a continuous function from $\mathbb{R}^{n+2} \times \Omega$ into $\mathbb{R}$. Note that for $\mathfrak{S}' := \{-1, 1\} \times \{0\}^k$ and $\Omega' := I \times \mathfrak{S}' \subseteq \Omega$, ($P_{\text{minimax}}$) is exactly retrieved as shown in the next example.

■ **Example 8.5** *For $n = 5$, **Problem** (P_{minimax}) is:*

$$\begin{array}{ll} \min\limits_{(\overline{a},\boldsymbol{a})\in\mathbb{R}^7} & \overline{a} \\ \text{s.t.} & (1, \sigma_0 1, \sigma_0 t, \dots, \sigma_0 t^5)(\overline{a}, a_0, a_1, \dots, a_5)^T \geqslant \sigma_0 f(t), \\ & \sigma_0 = \mp 1, \quad t \in I. \end{array} \tag{Example 3 (a)}$$

*while **Problem** (P'_{general}), assuming Horner evaluation is:*

$$\begin{array}{ll} \min\limits_{(\overline{a},\boldsymbol{a})\in\mathbb{R}^7} & \overline{a} \\ \text{s.t.} & (1, \sigma_0 + \sigma_1 u, (\sigma_0 + \sigma_1 u + \sigma_2 2u)t, \dots, \\ & (\sigma_0 + \sigma_1 u + \dots + \sigma_5 u)t^5)(\overline{a}, a_0, a_1, \dots, a_5)^T \geqslant \sigma_0 f(t), \\ & \sigma_0 = \mp 1, \sigma_1 = \mp 1, \dots, \sigma_5 = \mp 1, \quad t \in I. \end{array} \tag{Example 3 (b)}$$

In **Section 8.4** an exchange algorithm which solves **Problem** (P'_{general}) is presented. It can be seen as a generalization, in the above framework, of the Remez algorithm, which solves **Problem** (P_{minimax}). To prove its correctness, important discretization properties of linear SIP problems are recalled, closely following the survey **[228]**.

8.3.2 ▶ Duality and discretization for SIP

For a Problem (P), we denote respectively by $\text{val}(P)$ and $\text{Sol}(P)$, its optimal value and the set of its optimal solutions.

A discretization (P_m) of (P) for a set $\boldsymbol{\omega} = \{\omega_1, \ldots, \omega_m\} \subseteq \Omega$ is the following linear program:

$$\begin{array}{ll} \min\limits_{\boldsymbol{x} \in \mathbb{R}^d} & \boldsymbol{z}^T \boldsymbol{x} \\ \text{s.t.} & \boldsymbol{\alpha}(\omega_j)^T \boldsymbol{x} \geqslant c(\omega_j), \quad j = 1, \cdots m. \end{array} \tag{P_m}$$

Since the feasible set of (P) is included in the feasible set of (P_m), we have that $\text{val}(P_m) \leqslant \text{val}(P)$. The existence of a discretization (P_m) such that the equality holds is a particularly appealing feature of some linear SIPs since the solution of (P) may be obtained by the solution of (P_m) if we are able to find the corresponding set $\boldsymbol{\omega}$.

Definition 8.6 [228] *(P) is said to be* reducible *if there exists a discretization (P_m) defined by the subset $\{\omega_1, \ldots, \omega_m\} \subseteq \Omega$ such that* $\text{val}(P_m) = \text{val}(P)$.

The characterization of reducible SIP problems relies on the central notion of *duality* that rules the interplay between two optimization problems. This notion has its roots in the interrelations between a normed linear space and its topological dual. Let us define the continuous mapping $g : \boldsymbol{x} \mapsto g(\boldsymbol{x}, \cdot)$ from $\mathbb{R}^d$ to the Banach space of continuous functions $\mathcal{C}(\Omega)$, equipped with the uniform norm $\|h\|_\Omega = \sup_{\omega \in \Omega} |h(\omega)|$, then the topological dual of $\mathcal{C}(\Omega)$ is the space $\mathcal{C}(\Omega)^*$ of signed Borel measures μ over $(\Omega, \mathcal{B}(\mathbb{R}^{k+2}))$ [103, Section 21.5]. For a measure $\mu \in \mathcal{C}(\Omega)^*$, its support is the smallest closed subset Γ of Ω such that $|\mu|(\Omega \setminus \Gamma) = 0$. A positive measure μ is denoted by $\mu \succeq 0$. A classical example of a positive measure with discrete support is the Dirac measure of support $\{\omega_j\}$:

$$\delta_{\omega_j}(A) = \left\{ \begin{array}{lll} 0 & \text{if} & \omega_j \notin A, \\ 1 & \text{if} & \omega_j \in A. \end{array} \right. \qquad A \subseteq \Omega.$$

Defining the bilinear form pairing $\mathcal{C}(\Omega)$ and $\mathcal{C}(\Omega)^*$ by the duality bracket:

$$\langle h, \mu \rangle = \int_\Omega h(\omega) d\mu(\omega),$$

the *dual* problem (D) related to the *primal* problem (P) is:

$$\begin{array}{ll} \max\limits_{\mu \succeq 0} & \displaystyle\int_\Omega c(\omega) \mathrm{d}\mu(\omega) \\ \text{s.t.} & \displaystyle\int_\Omega \boldsymbol{\alpha}(\omega) \mathrm{d}\mu(\omega) = \boldsymbol{z}. \end{array} \tag{D}$$

The *weak duality*, that is $\text{val}(D) \leqslant \text{val}(P)$ always holds. **Problem (D)** is an LP problem defined in the space of positive measures which is hard to solve. By restricting the support of $\mu \succeq 0$ to $\{\omega_1, \ldots, \omega_m\}$, that is $\mu = \sum_{j=1}^m y_j \delta_{\omega_j}$ with $y_j \geqslant 0$, a discretized counterpart (D_m) of (D) is obtained:

$$\begin{array}{ll} \max\limits_{\substack{y_j \geqslant 0 \\ j \in [\![1,m]\!]}} & \displaystyle\sum_{j=1}^m c(\omega_j) y_j \\ \text{s.t.} & \displaystyle\sum_{j=1}^m y_j \boldsymbol{\alpha}(\omega_j) = \boldsymbol{z}, \end{array} \tag{D_m}$$

with $\text{val}(D_m) \leqslant \text{val}(D)$. It is important to note that the LP dual of the discretized problem (P_m) is exactly (D_m) which implies that $\text{val}(D_m) = \text{val}(P_m)$ (strong duality holds) provided that none of (P_m) or (D_m) is infeasible.

So far, under these mild assumptions, we have that $\text{val}(D_m) = \text{val}(P_m) \leqslant \text{val}(D) \leqslant \text{val}(P)$ and conditions for having only equalities (respectively reducibility and strong duality properties) may be obtained by using conjugate duality theory as developed in **[228,** Theorems 2.2, 2.3 and 3.2**]**.

Theorem 8.7 [228, Thm. 2.2, 2.3, 3.2**]** *Under the assumptions:*

A1 Ω *is a compact metric space,* $\boldsymbol{\alpha} : \Omega \to \mathbb{R}^d$ *and* $c : \Omega \to \mathbb{R}$ *are continuous functions;*

A2 $\text{val}(P)$ *is finite;*

A3 (Slater's condition): there exists $\boldsymbol{x}^\circ$ *such that:*

$$\boldsymbol{\alpha}(\omega)^T \boldsymbol{x}^\circ > c(\omega), \quad \text{for all } \omega \in \Omega;$$

A4 There exist $\omega_1, \ldots, \omega_d \in \Omega$ *with* $(\boldsymbol{\alpha}(\omega_1), \cdots, \boldsymbol{\alpha}(\omega_d))$ *linearly independent such that:*

$$\exists\ y_1, \ldots, y_d > 0, \quad \boldsymbol{z} = \sum_{j=1}^{d} y_j \boldsymbol{\alpha}(\omega_j),$$

the following statements are true:

(i) $\text{Sol}(P) \neq \varnothing$ *and bounded;*

(ii) $\text{Sol}(D) \neq \varnothing$ *and bounded;*

(iii) ***Problem*** (P) *is reducible to a* ***Problem*** (P_m) *with* $m \leqslant d$*;*

(iv) $\text{val}(P) = \text{val}(D) = \text{val}(P_m) = \text{val}(D_m)$.

Proposition 8.8 *Assumptions A1-A4 are satisfied for our* ***Problem*** (P'_{general}) *and therefore results (i)-(iv) of* ***Theorem 8.7*** *apply.*

Proof.

A1 By construction, our set Ω is a compact metric space and α and c are polynomials and therefore continuous on Ω;

A2 $\text{val}(P) = +\infty$ means that the primal problem (P'_{general}) is not feasible but $\boldsymbol{x} = (\max_{t \in I} |f(t)|, 0 \cdots, 0)$ is a feasible point for (P'_{general}), therefore $\text{val}(P) < +\infty$. In addition, $\text{val}(P) > -\infty$ since $\overline{a} \geqslant 0$ by construction and for all feasible points of (P'_{general});

A3 It may be easily deduced from the proof of A2 that $\boldsymbol{x}^\circ = (\max_{t \in I} |f(t)| + \varsigma, 0 \cdots, 0)$ is a strictly feasible point for (P'_{general}) for any $\varsigma > 0$;

A4 An instance for $\{\omega_1, \ldots, \omega_{n+2}\}$ is provided by **Algorithm** INIT in **Section 8.4**.

□

The fact that both (P) and (D) are reducible to a discretization of size at most d, allows for recasting the problem (P'_{general}) as the problem of finding the right discretization $\{\omega_1, \ldots, \omega_d\}$ such that item (iv) of **Theorem 8.7** applies and to solve the associated (P_m) and/or (D_m). This goal may be reached by tailoring the general exchange algorithm for semi-infinite linear programs presented in **[49]** to our specific case.

This algorithm can be seen as a generalization of the dual simplex algorithm for **Problem** (D). The main idea consists first in finding at each iteration ℓ, the solution $\boldsymbol{y}^{(\ell)}$ of $(D^{(\ell)}_{n+2})$, with $\boldsymbol{\omega}^{(\ell)} = \{\omega_j{}^{(\ell)}\}_{j=1}^{n+2}$. Such a solution is a feasible (but not necessarily optimal) point of the dual **Problem** (D). Moreover, the objective value $\mathbf{z}^T\mathbf{x}^{(\ell)}$ of $(P^{(\ell)}_m)$ and (P) for the instance $\boldsymbol{x}^{(\ell)} := (\overline{a}^{(\ell)}, \boldsymbol{a}^{(\ell)})$ is equal to the objective value of (D) for the instance $\boldsymbol{y}^{(\ell)}$[2]. Hence, either $\boldsymbol{x}^{(\ell)}$ is a feasible solution of **Problem** (P) by **Theorem 8.7**, or it is an infeasible point of **Problem** (P). In the latter case, one of these constraints is replaced by a new one, indexed by $\omega_*{}^{(\ell)}$, in an exchange step in order to increase the objective value of the dual and works towards primal feasibility.

8.4 | Iterative exchange algorithm

Now that the essential notions and properties of linear SIP have been given in the previous section, we give the exchange algorithm EvalApproxOptimize to solve **Problem** (P'_{general}) in **Section 8.4.1**. The correctness proofs are postponed in **Section 8.4.2**. After that, some insight into how to transpose this algorithm to the framework of relative error is given in **Section 8.4.3**.

8.4.1 ▶ The algorithms

Algorithm EvalApproxOptimize computes the degree-n best polynomial approximation with respect to both evaluation and approximation errors, i.e. it solves (P'_{general}) based on developments from **Section 8.3.2**. Regarding the main steps of the new algorithm, an analogy with Remez is as follows:
- Init provides *a good set of initial points.*
- At each step, SolvePrimal solves a linear system of equations (built w.r.t. the current set of points), where the variables are the polynomial coefficients.
- Then, FindNewIndex finds a new point where the *total error* is maximal.
- Finally, Exchange replaces one point from the current set with this new point.

However, when considering both errors, one can not only rely on the primal problem (coefficients reconstruction), but also needs the dual problem. This implies:
- Besides classical points, a combination of signs (signatures) is required at each step.
- Init and Exchange need the solution of the dual problem.

[2]The feasible set of (P) is included in the feasible set of $(P^{(\ell)}_m)$, for all ℓ.

A running example for this algorithm is given in **Section 8.5**.

Algorithm 8.3 EvalApproxOptimize(f, n, I, θ, τ)

Input: function f, $n \geqslant 0$, I, $\theta(\boldsymbol{a}, t)$ as in (8.5), $\tau > 0$.
Output: $(\overline{a}, \boldsymbol{a})$ solution of **Problem** (P) within accuracy τ.

▷ *Initialization*
$(\boldsymbol{\omega}^{(0)}, \boldsymbol{y}^{(0)}) \leftarrow$ Init(n, I)
$(\overline{a}^{(0)}, \boldsymbol{a}^{(0)}) \leftarrow$ SolvePrimal$(f, n, \theta, \boldsymbol{\omega}^{(0)})$
$({\omega_*}^{(0)}, \overline{a}_*^{(0)}) \leftarrow$ FindNewIndex$(f, n, I, \theta, \boldsymbol{a}^{(0)})$
$\ell \leftarrow 0$
▷ *Iterate while accuracy τ not reached*
while $\overline{a}_*^{(\ell)} / \overline{a}^{(\ell)} > 1 + \tau$ **do**
  $(\boldsymbol{\omega}^{(\ell+1)}, \boldsymbol{y}^{(\ell+1)}) \leftarrow$ Exchange$(n, \theta, \boldsymbol{\omega}^{(\ell)}, \boldsymbol{y}^{(\ell)}, \omega_*^{(\ell)})$
  $(\overline{a}^{(\ell+1)}, \boldsymbol{a}^{(\ell+1)}) \leftarrow$ SolvePrimal$(f, n, \theta, \boldsymbol{\omega}^{(\ell+1)})$
  $({\omega_*}^{(\ell+1)}, \overline{a}_*^{(\ell+1)}) \leftarrow$ FindNewIndex$(f, n, I, \theta, \boldsymbol{a}^{(\ell+1)})$
  $\ell \leftarrow \ell + 1$
end while
return $(\overline{a}^{(\ell)}, \boldsymbol{a}^{(\ell)})$

Algorithm 8.4 Init(n, I)

Input: $n \geqslant 0$, $I = [t_l, t_r]$.
Output: $\boldsymbol{\omega} \in \Omega^{n+2}$ and solution $\boldsymbol{y}$ of **Problem** (D_m).

▷ *Initialize with Chebyshev nodes and Remez constraints*
for j **in** $[\![1, n+2]\!]$ **do**
  $t_j \leftarrow \frac{t_l + t_r}{2} + \cos\left(\frac{(j-1)\pi}{n+1}\right) \frac{t_l - t_r}{2}$
  $\boldsymbol{\sigma}_j \leftarrow ((-1)^j, 0, \ldots, 0)$
  $\omega_j \leftarrow (t_j, \boldsymbol{\sigma}_j)$
end for
▷ *Compute dual solution*
Solve for $\boldsymbol{y}$ the linear system $\sum_{j=1}^{n+2} y_j \boldsymbol{\alpha}(\omega_j) = \boldsymbol{z}$
return $(\boldsymbol{\omega}, \boldsymbol{y})$

Algorithm 8.5 SolvePrimal$(f, n, \theta, \boldsymbol{\omega})$

Input: function f, $n \geqslant 0$, θ the evaluation error, $\boldsymbol{\omega} \in \Omega^{n+2}$.
Output: $(\overline{a}, \boldsymbol{a})$ solution of **Problem** (P_m) for $\boldsymbol{\omega}$.

1: Solve for $(\overline{a}, \boldsymbol{a}) \in \mathbb{R}^{n+2}$ the linear system:

$$\boldsymbol{\alpha}(\omega_j)^T(\overline{a}, \boldsymbol{a}) = c(\omega_j), \quad j \in [\![1, n+2]\!]$$

2: **return** $(\overline{a}, \boldsymbol{a})$

Algorithm 8.6 FindNewIndex$(f, n, I, \theta, \boldsymbol{a})$

Input: function f, $n \geqslant 0$, $I = [t_l, t_r]$, θ the evaluation error as in (8.5), coefficients $\boldsymbol{a} \in \mathbb{R}^{n+1}$.
Output: $(\omega_*, \overline{a}_*)$ with $\omega_* = (t_*, \boldsymbol{\sigma}_*) \in \Omega$

▷ *Compute maximal error in absolute value*
1: $t_* \leftarrow \underset{t_l \leqslant t \leqslant t_r}{\arg\max} |\boldsymbol{a}^T \boldsymbol{\pi}_0(t) - f(t)| + \sum_{i=1}^{k} |\boldsymbol{a}^T \boldsymbol{\pi}_i(t)|$
2: $\overline{a}_* \leftarrow \underset{t_l \leqslant t \leqslant t_r}{\max} |\boldsymbol{a}^T \boldsymbol{\pi}_0(t) - f(t)| + \sum_{i=1}^{k} |\boldsymbol{a}^T \boldsymbol{\pi}_i(t)|$
▷ *Reconstruct signature*
3: $\sigma_{*0} \leftarrow -\operatorname{sign}(\boldsymbol{a}^T \boldsymbol{\pi}_0(t_*) - f(t_*))$
4: $\sigma_{*i} \leftarrow -\operatorname{sign}(\boldsymbol{a}^T \boldsymbol{\pi}_i(t_*)), \quad i \in [\![1, k]\!]$
5: $\omega_* \leftarrow (t_*, \boldsymbol{\sigma}_*)$
6: **return** $(\omega_*, \overline{a}_*)$

Algorithm 8.7 Exchange$(n, \theta, \boldsymbol{\omega}, \boldsymbol{y}, \omega_*)$

Input: $n \geqslant 0$, θ the evaluation error, $\boldsymbol{\omega} \in \Omega^{n+2}$, dual solution $\boldsymbol{y} \in \mathbb{R}^{n+2}$, new index $\omega_* \in \Omega$.
Output: new set $\boldsymbol{\omega'} \in \Omega^{n+2}$ and dual solution $\boldsymbol{y'} \in \mathbb{R}^{n+2}$.

1: Solve for $\boldsymbol{\gamma} \in \mathbb{R}^{n+2}$ the linear system:

$$\sum_{j=1}^{n+2} \gamma_j \boldsymbol{\alpha}(\omega_j) = \boldsymbol{\alpha}(\omega_*)$$

▷ *Exiting index*
2: $j_0 \leftarrow \arg\min \left\{ \frac{y_j}{\gamma_j} \;\middle|\; \gamma_j > 0 \right\}$
▷ *Update dual solution*
3: $\tilde{y}_* \leftarrow \frac{y_{j_0}}{\gamma_{j_0}}$
4: $\tilde{y}_j \leftarrow y_j - \gamma_j \tilde{y}_*, \quad j \in [\![1, n+2]\!]$
5: $\{(\boldsymbol{\omega'_j}, y'_j)\} \leftarrow \{(\omega_j, \tilde{y}_j), j \in [\![1, n+2]\!] - \{j_0\} \cup \{*\}\}$,
6: **return** $(\boldsymbol{\omega'}, \boldsymbol{y'})$

Remark 8.9 *Although the formulation of* FindNewIndex *theoretically requires the values of f over the whole interval $[t_l, t_r]$, which would contradict a black-box approach, in practice* FindNewIndex *is implemented via a discretization of $[t_l, t_r]$, evaluating f on it, and then picking t_* among these grid points.*

8.4.2 ▶ Correctness proof

We focus now on the correctness of **Algorithm** EvalApproxOptimize, which is stated in **Theorem 8.16**. For this, one needs an assumption on the dual solution, which always holds in the Remez algorithm. It is not proven in our setting, but it never failed in practice.

Assumption 8.10 *At each iteration ℓ, the solution $\boldsymbol{y}^{(\ell)}$ of the dual discretized* ***Problem*** *$(D_{n+2}^{(\ell)})$ is an interior point, that is $y_j^{(\ell)} > 0$ for all $j \in [\![1, n+2]\!]$.*

Moreover, one needs preliminary correctness proofs of **Algorithms** Init, SolvePrimal, FindNewIndex, and Exchange.

Lemma 8.11 (Correctness of Init) *Init(n, I) computes $\boldsymbol{\omega} = \{\omega_j\}_{j=1}^{n+2} \in \Omega^{n+2}$ and $\boldsymbol{y} \in \mathbb{R}^{n+2}$ satisfying:*

- *$\{\boldsymbol{\alpha}(\omega_j)\}_{j=1}^{n+2}$ is a basis of $\mathbb{R}^{n+2}$;*
- *$\boldsymbol{y}$ is the optimal solution of* ***Problem*** *(D_m) for $\boldsymbol{\omega}$;*
- *$y_j > 0$ for all $j \in [\![1, n+2]\!]$.*

Note that **Algorithm** Init essentially initializes the problem with Chebyshev nodes for heuristic efficiency and signatures corresponding to the classical Remez algorithm, without the evaluation error term.

Proof. Let $\boldsymbol{A}(\boldsymbol{\omega})$ denote the $(n+2)$ square matrix whose columns are the $\boldsymbol{\alpha}(\omega_j)$. Since $\boldsymbol{\sigma_j} = \big((-1)^j, 0, \dots, 0\big)$, we have

$$\boldsymbol{A}(\boldsymbol{\omega}) = \begin{pmatrix} 1 & \dots & \dots & 1 \\ -1 & \dots & \dots & (-1)^{n+2} \\ -t_1 & \dots & \dots & (-1)^{n+2} t_{n+2} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ -t_1^n & \dots & \dots & (-1)^{n+2} t_{n+2}^n \end{pmatrix}.$$

First, we prove the existence of a feasible point $\boldsymbol{y}$ in **Problem** (D_m) for $\boldsymbol{\omega}$, that is $\boldsymbol{A}(\boldsymbol{\omega})\boldsymbol{y} = \boldsymbol{z}$ and $\boldsymbol{y} \geqslant 0$. From Farkas' lemma [26], if such a $\boldsymbol{y}$ does not exist, then there exists $\boldsymbol{x} = (\overline{a}, \boldsymbol{a}) \in \mathbb{R}^{n+2}$ s.t. $\boldsymbol{z}^T\boldsymbol{x} = \overline{a} < 0$ and $\boldsymbol{A}(\boldsymbol{\omega})^T\boldsymbol{x} \geqslant 0$, that is

$$\overline{a} + (-1)^j \boldsymbol{a}^T \boldsymbol{\pi_0}(t_j) \geqslant 0, \qquad j \in [\![1, n+2]\!].$$

Since $\overline{a} < 0$, this implies that $\operatorname{sign}(\boldsymbol{a}^T\boldsymbol{\pi_0}(t_j)) = (-1)^j$. But $\boldsymbol{a}^T\boldsymbol{\pi_0}(t)$ is a polynomial of degree at most n, hence it cannot strictly change signs $n+2$ times. Consequently, **Problem** (D_m) has a feasible point $\boldsymbol{y}$.

Now, suppose that the columns of $\boldsymbol{A}(\boldsymbol{\omega})$ are not linearly independent, or that $y_j = 0$ for some j. Both cases imply that there exists $J \subset [\![1, n+2]\!]$ of size $n+1$, and $\tilde{\boldsymbol{y}} \in \mathbb{R}^{n+1}$ s.t. $\sum_{j \in J} \tilde{y}_j \boldsymbol{\alpha}(\omega_j) = \boldsymbol{z}$. In particular, by canceling the first component, the family $\{\boldsymbol{\pi_0}(t_j)\}_{j \in J}$ is linearly dependent. But the Vandermonde determinant of this system cannot vanish since the t_j are pairwise distinct. Therefore, $\{\boldsymbol{\alpha}(\omega_j)\}_{j=1}^{n+2}$ is a basis of $\mathbb{R}^{n+2}$, and $y_j > 0$ for all j.

Finally, since $\boldsymbol{A}(\boldsymbol{\omega})$ is invertible, $\boldsymbol{y}$ is the unique optimal feasible point of (D_m). □

Lemma 8.12 (Correctness of SolvePrimal) *If $\{\boldsymbol{\alpha}(\omega_j)\}_{j=1}^{n+2}$ is a basis of $\mathbb{R}^{n+2}$ and Problem (D_m) for $\boldsymbol{\omega}$ is feasible, then* SolvePrimal$(f, n, \theta, \boldsymbol{\omega})$ *computes the optimal solution $\boldsymbol{x} = (\overline{a}, \boldsymbol{a})$ of Problem (P_m).*

Proof. **Algorithm** SolvePrimal computes the solution $\boldsymbol{x} \in \mathbb{R}^{n+2}$ of $\boldsymbol{\alpha}(\omega_j)^T \boldsymbol{x} = c(\omega_j)$ for $j \in [\![1, n+2]\!]$. We show that $\boldsymbol{x}$ is the optimal solution of **Problem** (P_m) for $\boldsymbol{\omega}$.

Let $\tilde{\boldsymbol{x}}$ be any feasible point in (P_m). Since the dual **Problem** (D_m) for $\boldsymbol{\omega}$ is feasible, there exists $\boldsymbol{y} \geqslant 0$ s.t. $\boldsymbol{z} = \sum_{j=1}^{n+2} y_j \boldsymbol{\alpha}(\omega_j)$. Then

$$\boldsymbol{z}^T \tilde{\boldsymbol{x}} = \sum_{j=1}^{n+2} y_j \boldsymbol{\alpha}(\omega_j)^T \tilde{\boldsymbol{x}} \;\geqslant\; \sum_{j=1}^{n+2} y_j c(\omega_j) = \sum_{j=1}^{n+2} y_j \boldsymbol{\alpha}(\omega_j)^T \boldsymbol{x} = \boldsymbol{z}^T \boldsymbol{x},$$

thereby establishing optimality of $\boldsymbol{x}$. □

Lemma 8.13 (Correctness of FindNewIndex) *Given $\boldsymbol{x} = (\overline{a}, \boldsymbol{a})$,* FindNewIndex$(f, n, I, \theta, \boldsymbol{a})$ *computes ω_* and $\overline{a}_*$ corresponding to the most violated constraint:*

$$\omega_* = \arg\max_{\omega \in \Omega} \left(c(\omega) - \boldsymbol{\alpha}(\omega)^T \boldsymbol{x} \right),$$
$$\overline{a}_* - \overline{a} = \max_{\omega \in \Omega} \left(c(\omega) - \boldsymbol{\alpha}(\omega)^T \boldsymbol{x} \right).$$

Proof. We have

$$\begin{aligned} \max_{\omega \in \Omega} \left(c(\omega) - \boldsymbol{\alpha}(\omega)^T \boldsymbol{x} \right) &= \max_{\omega = (t, \boldsymbol{\sigma}) \in \Omega} \left(\sigma_0 \left(f(t) - \boldsymbol{a}^T \boldsymbol{\pi_0}(t) \right) - \sum_{i=1}^{k} \sigma_i \boldsymbol{a}^T \boldsymbol{\pi_i}(t) \right) - \overline{a} \\ &= \max_{t_l \leqslant t \leqslant t_r} \left(|\boldsymbol{a}^T \boldsymbol{\pi_0}(t) - f(t)| + \sum_{i=1}^{k} |\boldsymbol{a}^T \boldsymbol{\pi_i}(t)| \right) - \overline{a}. \end{aligned}$$

Therefore, by computing t_* (line 1) and $\boldsymbol{\sigma_*}$ (lines 3-4), **Algorithm** FindNewIndex ensures that $\omega_* := (t_*, \boldsymbol{\sigma_*})$ is the index of the most violated constraint, with

$$c(\omega_*) - \boldsymbol{\alpha}(\omega_*)^T \boldsymbol{x} = \overline{a}_* - \overline{a} \geqslant 0.$$

□

Lemma 8.14 (Correctness of Exchange) *If $\{\boldsymbol{\alpha}(\omega_j)\}_{j=1}^{n+2}$ is a basis of $\mathbb{R}^{n+2}$ and $\boldsymbol{y}$ the optimal solution of Problem (D_m) for $\boldsymbol{\omega}$, then* Exchange$(n, \theta, \boldsymbol{\omega}, \boldsymbol{y}, \omega_*)$ *computes new $\boldsymbol{\omega'} \in \Omega^{n+2}$ and $\boldsymbol{y'} \in \mathbb{R}^{n+2}$ such that:*

- $\boldsymbol{\omega'} = \{\omega_j\}_{j \in J} \cup \{\omega_*\}$ *for a subset $J \subset [\![1, n+2]\!]$ of size $n+1$;*

- $\{\boldsymbol{\alpha}(\omega_j)\}_{j=1}^{n+2}$ *is a basis of* $\mathbb{R}^{n+2}$;
- $\boldsymbol{y'}$ *is the optimal solution of* ***Problem*** (D_m) *for* $\boldsymbol{\omega'}$.

Proof. In order to increase the objective value in the dual **Problem** (D), a measure supported on $\boldsymbol{\omega} \cup \{\omega_*\}$ is looked for, requiring nonnegative coefficients $\tilde{y}_j$ for $j \in [\![1, n+2]\!] \cup \{*\}$ s.t.

$$\boldsymbol{z} = \sum_{j=1}^{n+2} \tilde{y}_j \boldsymbol{\alpha}(\omega_j) + \tilde{y}_* \boldsymbol{\alpha}(\omega_*) = \sum_{j=1}^{n+2} (\tilde{y}_j + \gamma_j \tilde{y}_*)\, \boldsymbol{\alpha}(\omega_j),$$

while maximizing $\tilde{y}_*$. But $\{\boldsymbol{\alpha}(\omega_j)\}_{j=1}^{n+2}$ being a basis of $\mathbb{R}^{n+2}$ implies

$$\tilde{y}_j + \gamma_j \tilde{y}_* = y_j, \qquad j \in [\![1, n+2]\!].$$

The nonnegativity constraints on the $\tilde{y}_j$ induces the choice of the exiting index ω_{j_0} (line 2) and the values of the $\tilde{y}_j$ (lines 3-4). Note that the first line of the linear system in line 1 says that the coefficients γ_j sum to 1. Hence, at least one of them is strictly positive, so that j_0 exists (line 2), though it is not necessarily unique.

Finally, $\{\boldsymbol{\alpha}(\omega_j)\}_{j \in [\![1,n+2]\!] - \{j_0\} \cup \{*\}}$ remains a basis of $\mathbb{R}^{n+2}$ since $\gamma_{j_0} \neq 0$, that is $\boldsymbol{\alpha}(\omega_*)$ is not in the linear subspace spanned by $\{\boldsymbol{\alpha}(\omega_j)\}_{j \in [\![1,n+2]\!] - \{j_0\}}$. □

In addition, the following lemma proves that the *discretized* error increases at each iteration.

Lemma 8.15 *The total error* $\overline{a}^{(\ell)}$ *computed over the discrete set* $\boldsymbol{\omega}^{(\ell)}$ *increases at each iteration.*

Proof. Let $\tilde{y}_*^{(\ell)}$ and $\gamma_j^{(\ell)}$ denote the variables $\tilde{y}_*$ and γ_j for $j \in [\![1, n+2]\!]$ in **Algorithm** EXCHANGE$(n, \theta, \boldsymbol{\omega}^{(\ell)}, \boldsymbol{y}^{(\ell)}, \omega_*^{(\ell)})$. By strong duality in linear programming, $\overline{a}^{(\ell)}$ is also the objective value of the optimal solution $\boldsymbol{y}^{(\ell)}$ in the discretized dual problem (D_{n+2}) for $\boldsymbol{\omega}^{(\ell)}$. Hence, by writing

$$\begin{aligned}
\overline{a}^{(\ell)} &= \sum_{j=1}^{n+2} y_j^{(\ell)} c(\omega_j^{(\ell)}), \quad \text{and} \\
\overline{a}^{(\ell+1)} &= \sum_{j=1}^{n+2} y_j^{(\ell+1)} c(\omega_j^{(\ell+1)}) \\
&= \sum_{j=1}^{n=2} \left(y_j^{(\ell)} - \gamma_j^{(\ell)} \tilde{y}_*^{(\ell)} \right) c(\omega_j^{(\ell)}) + \tilde{y}_*^{(\ell)} c(\omega_*^{(\ell)}),
\end{aligned}$$

we have

$$\begin{aligned}
\overline{a}^{(\ell+1)} - \overline{a}^{(\ell)} &= \tilde{y}_*^{(\ell)} \left(c(\omega_*^{(\ell)}) - \sum_{j=1}^{n+2} \gamma_j^{(\ell)} c(\omega_j^{(\ell)}) \right) \\
&= \tilde{y}_*^{(\ell)} \left(c(\omega_*^{(\ell)}) - \sum_{j=1}^{n+2} \gamma_j^{(\ell)} \boldsymbol{\alpha}(\omega_j^{(\ell)})^T \boldsymbol{x}^{(\ell)} \right) \\
&= \tilde{y}_*^{(\ell)} (c(\omega_*^{(\ell)}) - \boldsymbol{\alpha}(\omega_*^{(\ell)})^T \boldsymbol{x}^{(\ell)}) \geqslant 0,
\end{aligned}$$

because $\tilde{y}_*^{(\ell)} \geqslant 0$ and the constraint $\boldsymbol{\alpha}(\omega_*^{(\ell)})^T \boldsymbol{x}^{(\ell)} \geqslant c(\omega_*^{(\ell)})$ is violated at the beginning of iteration ℓ. □

This finally leads us to the central theorem of this section.

Theorem 8.16 *Let f be a continuous function over an interval $I = [t_l, t_r]$, a degree $n \geqslant 0$, a linearized evaluation error bound θ and a tolerance parameter $\tau > 0$. Under **Assumption 8.10**, EvalApproxOptimize(f, n, I, θ, τ) terminates and returns a degree-n polynomial approximation for f with a total error ε (approximation and evaluation) satisfying:*

$$\varepsilon^* \leqslant \varepsilon \leqslant (1+\tau)\varepsilon^*, \tag{8.6}$$

where ε^* is the total error of the best degree-n polynomial approximation of f.

Proof. • It is proven by induction that the following properties hold at each iteration $\ell \geqslant 0$:

(i) $\{\boldsymbol{\alpha}(\omega_j^{(\ell)})\}_{j=1}^{n+2}$ is a basis of $\mathbb{R}^{n+2}$;

(ii) $\boldsymbol{y}^{(\ell)}$ is the optimal solution of **Problem** (D_m) for $\boldsymbol{\omega}^{(\ell)}$;

(iii) $\boldsymbol{x}^{(\ell)}$ is the optimal solution of **Problem** (P_m) for $\boldsymbol{\omega}^{(\ell)}$;

(iv) $\omega_*^{(\ell)} = \arg\max_{\omega\in\Omega} \left(c(\omega) - \boldsymbol{\alpha}(\omega)^T \boldsymbol{x}^{(\ell)}\right)$;

For $\ell = 0$, Init(n, I) returns $\boldsymbol{\omega}^{(0)}$, $\boldsymbol{y}^{(0)}$ satisfying (i) and (ii). Then SolvePrimal$(f, n, \theta, \boldsymbol{\omega}^{(0)})$ computes $\boldsymbol{x}^{(0)} = (\overline{a}^{(0)}, \boldsymbol{a}^{(0)})$ satisfying (iii). Finally, FindNewIndex$(f, n, I, \theta, \boldsymbol{a}^{(0)})$ gives $\omega_*^{(0)}$, $\overline{a}_*^{(0)}$ satisfying (iv).

For the inductive step, Exchange$(n, \theta, \boldsymbol{\omega}^{(\ell)}, \boldsymbol{y}^{(\ell)}, \omega_*^{(\ell)})$ computes $\boldsymbol{\omega}^{(\ell+1)}$, $\boldsymbol{y}^{(\ell+1)}$ satisfying (i) and (ii), by induction hypothesis on $\boldsymbol{\omega}^{(\ell)}$, $\boldsymbol{y}^{(\ell)}$, $\omega_*^{(\ell)}$. Then, SolvePrimal$(f, n, \theta, \boldsymbol{\omega}^{(\ell+1)})$ and FindNewIndex$(f, n, I, \theta, \boldsymbol{a}^{(\ell+1)})$ compute $\boldsymbol{x}^{(\ell+1)}$, $\omega_*^{(\ell+1)}$, $\overline{a}_*^{(\ell+1)}$ satisfying (iii) and (iv).

• Moreover, at each iteration ℓ, we have $\overline{a}^{(\ell)} \leqslant \varepsilon^* \leqslant \overline{a}_*^{(\ell)}$. Indeed, $\boldsymbol{x}^{(\ell)}$ is the optimal solution of the discretized **Problem** (P_m) for $\boldsymbol{\omega}^{(\ell)}$, whose objective value $\overline{a}^{(\ell)}$ is less or equal to the optimal value ε^* of **Problem** (P). On the other side, $\overline{a}_*^{(\ell)}$ is the total error of degree-n polynomial $\boldsymbol{a}^{(\ell)T}\boldsymbol{\pi}_{\mathbf{0}}(t)$ and therefore, it is greater or equal to the optimal error ε^*. In addition, **Lemma 8.15** proves $\overline{a}^{(\ell)} \leqslant \overline{a}^{(\ell+1)}$.

• Finally, the convergence of this iterative process is proved by **[49, Theorem 2.1]**, relying on **Assumption 8.10**. Hence, **Algorithm** EvalApproxOptimize terminates at some iteration ℓ, with $\overline{a}_*^{(\ell)} \leqslant (1+\tau)\overline{a}^{(\ell)}$, yielding the enclosure **(8.6)**. □

8.4.3 ▶ Optimizing with the relative error

When considering the relative error in place of the absolute error, **Problem** (P_{general}) is replaced by

$$\min_{\substack{a_i\in\mathbb{R},\\ i\in[\![0,n]\!]}} \max_{t\in I} \left(\frac{|f(t) - p(t)| + |\tilde{p}(t) - p(t)|}{|f|} \right) \tag{$P^{\text{rel}}_{\text{general}}$}$$

where f is assumed to be strictly positive over I. This is equivalent to multiplying the error variable $\overline{a}$ by f in **Problem** (P), yielding the following new linear SIP formulation

$$\begin{array}{ll} \min\limits_{\boldsymbol{x}\in\mathbb{R}^d} & \boldsymbol{z}^T\boldsymbol{x} \\ \text{s.t.} & \tilde{\boldsymbol{\alpha}}(\omega)^T\boldsymbol{x} \geqslant c(\omega), \quad \omega \in \Omega, \end{array} \tag{P^{rel}}$$

with

$$\boldsymbol{x} = (\overline{a}, \boldsymbol{a}) \in \mathbb{R}^{n+2},\ \boldsymbol{z} = (1, 0, \ldots, 0) \in \mathbb{R}^{n+2},$$

$$\tilde{\boldsymbol{\alpha}}(t, \sigma_0, \ldots, \sigma_k) = (f(t), \sigma_0 \boldsymbol{\pi_0}^T(t) + \sum_{i=1}^{k} \sigma_i \boldsymbol{\pi_i}^T(t))^T \in \mathbb{R}^{n+2},$$

$$\mathfrak{S} = \{-1, 0, 1\}^{k+1},\ \omega = (t, \sigma_0, \ldots, \sigma_k) \in \Omega := I \times \mathfrak{S}.$$

Therefore, replacing $\boldsymbol{\alpha}(\omega)$ by $\tilde{\boldsymbol{\alpha}}(\omega)$ in EvalApproxOptimize and its subroutines provides an algorithm to compute the best degree-n polynomial w.r.t. both approximation and evaluation errors in relative setting.

8.5 Examples and conclusion

Algorithm EvalApproxOptimize is firstly illustrated by a tutorial example for Airy special function. Then approximations with binary64 coefficients of arcsin are presented.

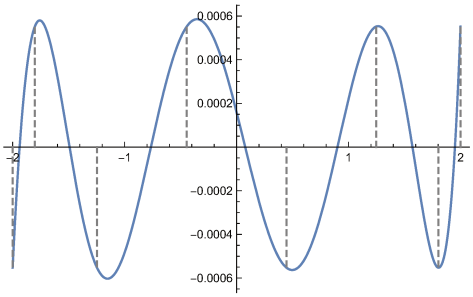
8.5.1 ▶ Airy function

The Airy function Ai is a special function frequently used in theoretical physics. In particular, some applications require to compute $\mathrm{Ai}(t)$ for possibly large negative values of t, where the function exhibits a highly oscillatory behavior (see **Figure 8.4a**). However, contrary to elementary functions, there exists no simple argument reduction for Ai. Therefore, one polynomial approximation is needed for each interval of the domain subdivision, and these intervals cannot be assumed to be small. Hence, controlling the evaluation error is essential.

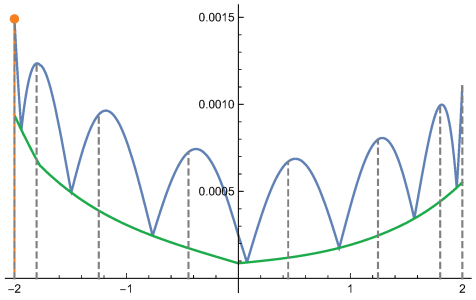
We consider the function Ai over $I = [-2, 2]$, approximated by a polynomial of degree $n = 6$ and evaluated using the Horner scheme with $u = 2^{-12}$. The terms $\{\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_7\}$ defining the evaluation error θ are given in **Example 8.3**. We fix a tolerance $\tau = 0.01$.

At iteration 0 (**Figure 8.1**), the points $t_j^{(0)}$ are initialized with the Chebyshev nodes and the signatures $\boldsymbol{\sigma_j}^{(0)}$ define a Remez-like system of linear equations on the coefficients of the polynomial (**Figure 8.1d**). Its solution $\boldsymbol{x}^{(0)} = (\overline{a}^{(0)}, \boldsymbol{a}^{(0)})$ defines a polynomial $p^{(0)}(t) = \boldsymbol{a}^{(0)T}\boldsymbol{\pi_0}(t)$, whose approximation error is depicted in **Figure 8.1a**. It exhibits quasi-equioscillations indicating that $\boldsymbol{p}^{(0)}$ is rather close to the degree-6 minimax approximation of Ai over I. However, the total error is more important near -2 and 2 (**Figure 8.1b**), due to the evaluation depicted in green. In particular, the algorithm detects the maximum error at $t_*^{(0)} = -2$ (in orange). Note that $t_1^{(0)}$ was already equal to -2, but $\omega_1^{(0)} \neq \omega_*^{(0)}$ since the signatures are different. To perform the exchange, the dual solution is needed (**Figure 8.1c**). It is a positive combination of Dirac measures supported on the finite set $\boldsymbol{\omega}^{(0)}$.

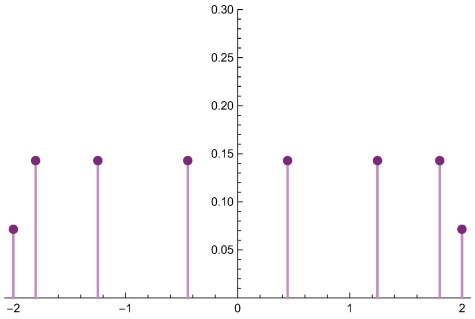
Moving forward to iteration 6 (**Figure 8.2**), the total error is more balanced, though still not optimal. Both the signatures and the approximation error are now completely different from the Remez solution.



(a) Approximation error



(b) Total error

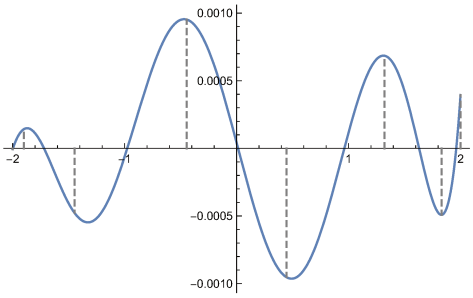


(c) Dual solution

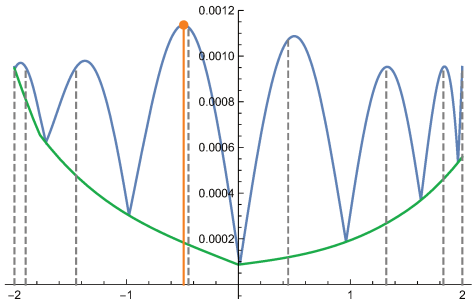
t_j	σ_{j0}	σ_{j1}	σ_{j2}	σ_{j3}	σ_{j4}	σ_{j5}	σ_{j6}	σ_{j7}
-2.	+	o	o	o	o	o	o	o
-1.803	-	o	o	o	o	o	o	o
-1.247	+	o	o	o	o	o	o	o
-0.445	-	o	o	o	o	o	o	o
0.445	+	o	o	o	o	o	o	o
1.247	-	o	o	o	o	o	o	o
1.802	+	o	o	o	o	o	o	o
2.	-	o	o	o	o	o	o	o

(d) Points and signatures

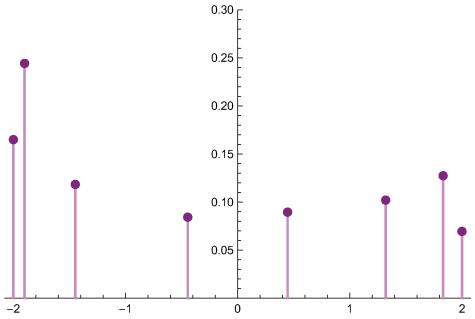
■ **Figure 8.1**: Approximation of Ai over $[-2, 2]$: iteration 0



(a) Approximation error



(b) Total error

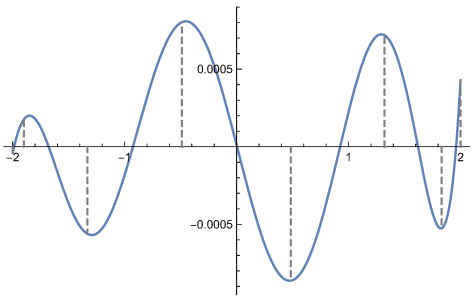


(c) Dual solution

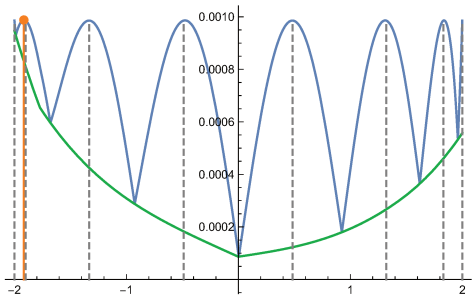
t_j	σ_{j0}	σ_{j1}	σ_{j2}	σ_{j3}	σ_{j4}	σ_{j5}	σ_{j6}	σ_{j7}
-2.	+	-	-	+	+	+	+	-
-1.9	-	-	-	+	+	+	+	-
-1.448	+	-	-	+	+	+	-	-
-0.445	-	o	o	o	o	o	o	o
0.445	+	o	o	o	o	o	o	o
1.32	-	-	+	+	-	-	+	-
1.832	+	-	-	+	-	-	+	-
2.	-	-	-	+	-	-	+	-

(d) Points and signatures

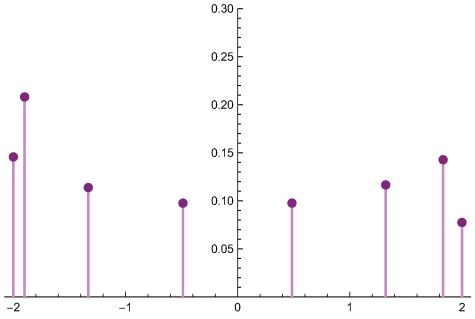
■ **Figure 8.2**: Approximation of Ai over $[-2, 2]$: iteration 6



(a) Approximation error



(b) Total error



(c) Dual solution

t_j	σ_{j0}	σ_{j1}	σ_{j2}	σ_{j3}	σ_{j4}	σ_{j5}	σ_{j6}	σ_{j7}
-2.	+	-	-	+	+	+	+	-
-1.9	-	-	-	+	+	+	+	-
-1.332	+	-	-	+	+	+	-	-
-0.488	-	-	-	+	+	+	-	-
0.484	+	-	+	+	-	-	+	-
1.32	-	-	+	+	-	-	+	-
1.832	+	-	-	+	-	-	+	-
2.	-	-	-	+	-	-	+	-

(d) Points and signatures

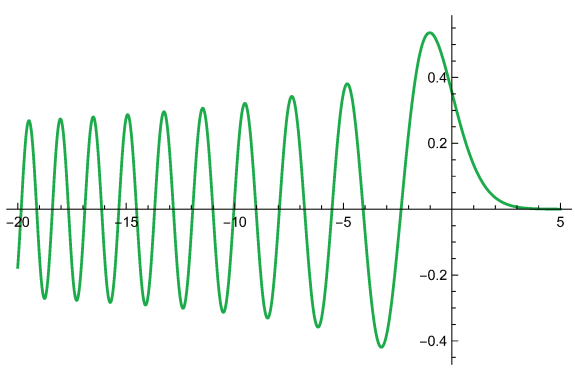
■ **Figure 8.3**: Approximation of Ai over $[-2, 2]$: iteration 9

Eventually, the algorithm stops at iteration 9 (**Figure 8.3**). Indeed, the maximum total error $\overline{a}_*^{(9)}$ (in orange) is less than 1% higher than the error $\overline{a}^{(9)}$ over the discrete set $\boldsymbol{\omega}^{(9)}$. Note that the total error reaches its maximum at $n+2=8$ points. This became possible by unbalancing the approximation error, namely reducing the amplitude of the oscillations near -2 and 2, at the cost of higher oscillations in the middle of I.

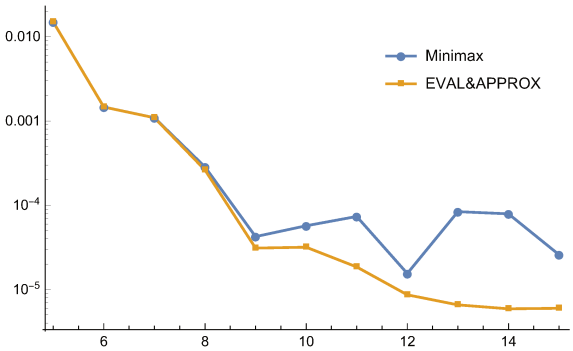
Now, we consider the approximation of the Airy function over $I=[-4,0]$ with polynomials evaluated using the Horner scheme with binary32 floating-point numbers, that is, single precision ($u=2^{-24}$). **Figure 8.4b** shows the total error (approximation and evaluation) of the degree-n minimax polynomial (in blue), in function of n. Specifically, the total error starts to decrease when n increases, thanks to the improvement of the approximation error. However, at some point (here $n=9$), the total error starts again to increase, due to the evaluation error of higher degree polynomials. On the contrary, the evaluation error of polynomials obtained with EvalApproxOptimize (in yellow) continues decreasing to some asymptotic threshold (around $5\cdot 10^{-6}$). In such a case, reducing the evaluation error by using a higher degree can be more efficient than increasing the floating-point precision.

8.5.2 ▶ Arcsine function

Consider $f=\arcsin$, over the interval $I=[0.75,1]$. This example is particularly insightful because f is ill-conditioned over I, and also because its values are close to 1, so absolute or relative error treatment is similar. Firstly, assume that argument reduction techniques are not available (this can be the case for any ill-conditioned function, known only through sampling



(a) Airy function Ai



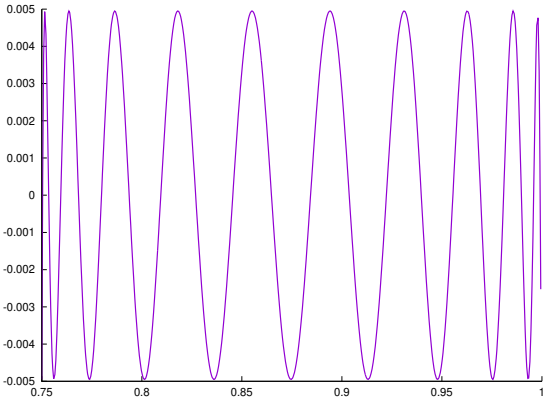
(b) Total error over $[-4, 0]$ for Minimax and EvalApproxOptimize, in function of degree

■ **Figure 8.4**: Approximation of Ai

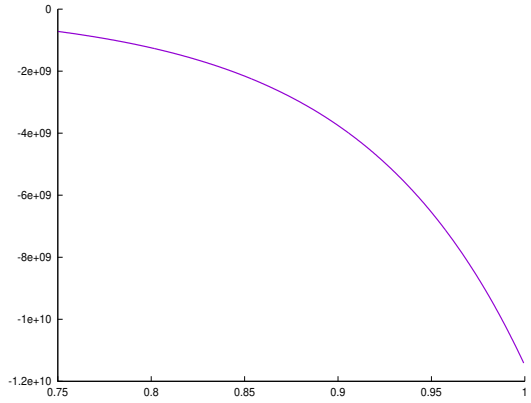
via a black-box approach), operations are in binary64 and an approximation polynomial of degree 20 with binary64 coefficients is searched for.

Figure 8.5(a) shows the absolute approximation error between arcsin and the best approximation polynomial p_{Remez} with *real* coefficients, for which $\max_I |f - p_{\text{Remez}}| \leqslant 0.00496$. Then, the coefficients of p_{Remez} are directly rounded to binary64, which results in $\max_I |f - \text{roundCoeff}(p_{\text{Remez}})| \simeq 0.004961.15 \cdot 10^{10}$ (cf. **Figure 8.5**(b)). This is due to the high magnitude of the coefficients. To search for better coefficients, one could use the proficient FPMinimax routine of Sollya [55], which optimizes on the coefficient space of representable FP numbers. The approximation error in this case is shown in **Figure 8.5**(c) and one has $\max_I |f - p_{\text{FPMinimax}})| \simeq 1.58$. Finally, in **Figure 8.5**(d) we provide the plot of the total error between f and the obtained EvalApproxOpt polynomial (with binary64 coefficients), which provides the best bound in this case $\max_I |f - p_{\text{E\&A}})| \simeq 0.0081$. One can then actually prove with Gappa [66] that the Horner scheme evaluation of $p_{\text{E\&A}}$ has an absolute error less than 0.00035.

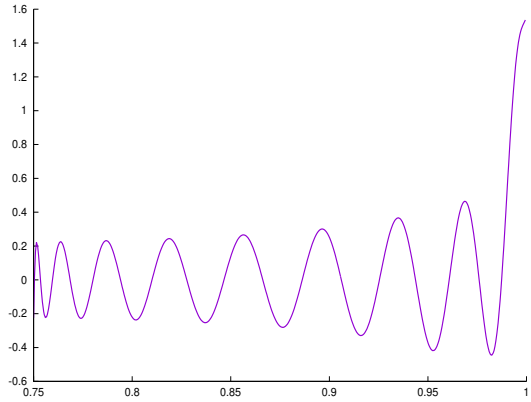
Secondly, suppose that I is re-centered to $I_c = [-0.125, 0.125]$ and the other parameters remain unchanged. Thus, the function to be approximated is $f = \arcsin(x + c)$, where c is the middle point of I. In this case, it turns out that all the methods presented above provide almost the same accuracy. In **Figure 8.6**(a), the plot of the absolute error between f and p_{Remez} is given: $\max_{I_c} |f - p_{\text{Remez}}| \leqslant 0.00496$; at the drawing scale, the plot is the same for the other errors and one has that $\max_{I_c} |f - \text{roundCoeff}(p_{\text{Remez}})|$, $\max_{I_c} |f - p_{\text{FPMinimax}})|$ and $\max_{I_c} |f - p_{\text{E\&A}})|$ are all very close in magnitude $\simeq 0.00496$. Similarly, one can prove with Gappa that the Horner scheme evaluation has an absolute error of $7.16227 \cdot 10^{-11}$ for every above-mentioned polynomial. Hence in this case, our algorithm does not improve the error bound. Yet, it shows (at least numerically) that if only binary64 computations and a Horner scheme are used, there is no other polynomial which performs better w.r.t. both evaluation and approximation errors. For completeness, in **Figure 8.6**(b) we show that FPMinimax and EvalApproxOpt polynomials are different. However, this difference is very small and this does not influence the number of correct bits provided $-\log_2(0.00496) \simeq 7.65$. A further study is to be performed on various functions and intervals in order to assess the practical improvement that can be expected from our method in general.



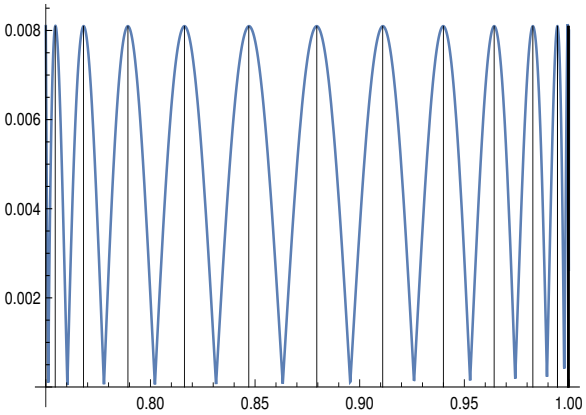
(a) $f - p_{\text{Remez}}$.



(b) $f - \text{roundCoeff}(p_{\text{Remez}})$.

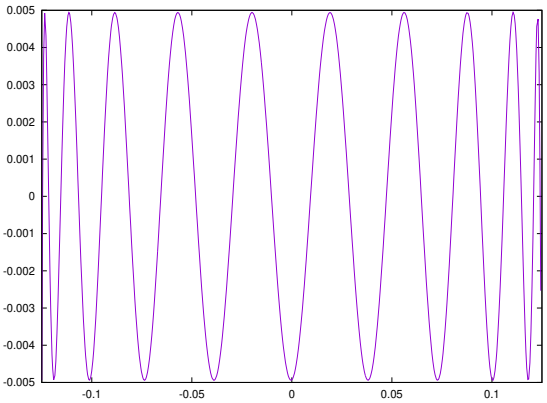


(c) $f - p_{\text{FPMinimax}}$.

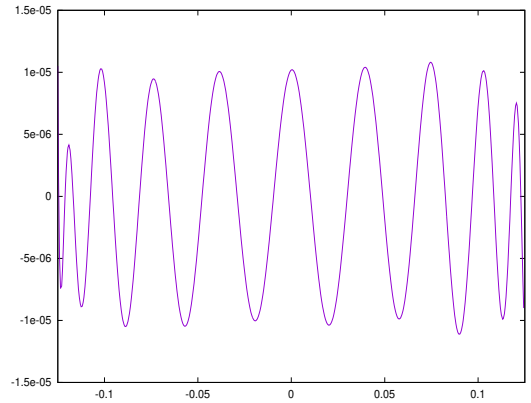


(d) EvalApproxOpt absolute error.

■ **Figure 8.5**: Error plots for different approximation polynomials of degree 20 for $f = \arcsin$ over $I = [0.75, 1]$.



(a) $f - p_{\text{Remez}}$



(b) $p_{\text{FPMinimax}} - p_{\text{E\&A}}$.

■ **Figure 8.6**: Error plots for different approximation polynomials of degree 20 for $f = \arcsin(x + c)$ over $I_c = [-0.125, 0.125]$.

8.5.3 ▶ Some comments

Finally, we mention that when other argument reduction techniques exist, and when the evaluation error is not an issue (very small intervals around zero), the FPMinimax method still provides better tuned FP coefficients. So this opens the question for several future extensions. A mixed-integer linear programming problem could be formulated in the provided optimization framework. However, a similar exchange procedure in this case is not obvious. Concerning precisions of the coefficients and operations, they can be variable, as mentioned in **Section 8.2**, but a more detailed study is needed to eventually take into account higher order error terms for the error estimation formula. The polynomial coefficients stay linear in such a formula, so the algorithm presented can be straightforwardly used in such a case. In addition, this formula directly allows for the estimation of evaluation errors for other numerical schemes and eventually polynomial bases [14] for which a practical study is necessary.

On a Moment Problem with Holonomic Functions

9

Quand on voit ce qu'on voit, que l'on entend ce qu'on entend et que l'on sait ce que qu'on sait, on a raison de penser ce qu'on pense.

— Pierre Dac, L'Os à moelle

Reconstructing algebraic data, such as polytope or more complex volume boundaries, from moment measurements is an essential requirement for computer tomography and shape recognition, with applications to, e.g., medical imaging or geophysics. Many algorithms resorting to optimization, numerical analysis or statistics, for instance, were developed toward this aim, as discussed in **Section 9.1**. In particular, Lasserre and Putinar **[150]** proposed an exact reconstruction algorithm for the algebraic support of the Lebesgue measure, or of measures with density equal to the exponential of a known polynomial. Their approach relies on linear recurrences for the moments obtained using Stokes theorem, whence a strong, yet not fully exploited connection with the D-finite setting presented in **Chapter 2**.

Guided by the intuition that holonomicity could shed a new light on that work, Mioara Joldes, Jean-Bernard Lasserre and I worked together toward extending this method to measures with unknown holonomic densities and unknown support with real algebraic boundary. The article that we wrote, "On Moment Problem with Holonomic Functions" **[39]**, will be published in the proceedings of the *44th International Symposium on Symbolic and Algebraic Computation* (ISSAC 2019).

More specifically, our work, summarized in this chapter, consists in two contributions. First, in the framework of holonomic distributions (i.e. they satisfy a holonomic system in the sense of distributions, see **Sections 2.1.2** and **9.2**), an alternate method to creative telescoping is proposed in **Section 9.3** for computing linear recurrences for the moments. When the coefficients of a polynomial vanishing on the support boundary are given as parameters, the obtained recurrences have the advantage of remaining linear with respect to them. Second, and based on this property, an efficient reconstruction method is explained in **Section 9.4**. Given a finite number of numerically computed moments for a measure with holonomic density, and assuming a real algebraic boundary for the support, we propose an algorithm for solving the inverse problem of obtaining both the coefficients of a polynomial vanishing on the boundary and those of the polynomials involved in the holonomic operators which annihilate the density.

9.1 Overview of algebraic techniques for reconstruction from moments

The structure of moments of algebraic data is a central question in various reconstruction algorithms, appearing as part of a broad field of inverse problems [133]. We refer to [150] and references therein for various shape reconstruction from their moments of polyhedra [98, 94], planar quadrature domains [76], sublevel sets of homogeneous polynomials [149], together with more applied studies of computerized tomography [180].

In this chapter, we focus on the structure of moments of holonomic distributions, together with associated inverse problems. It can be seen as a computer algebra-based extension of [150], where the approach was mainly based on techniques recently developed in polynomial optimization [148], which are at the interface between real algebraic geometry, moment problems and polynomial optimization.

Notations. Let n be a positive integer for the ambient space $\mathbb{R}^n$, whose canonical basis is denoted by $(\boldsymbol{e_1}, \dots, \boldsymbol{e_n})$. Let $\mathbb{K}[\boldsymbol{x}]$ be the ring of polynomials in the variables $\boldsymbol{x} = (x_1, \dots, x_n)$ over a real finite computable extension of $\mathbb{Q}$, and let $\mathbb{K}[\boldsymbol{x}]_d$ be the vector space of polynomials of total degree at most d. For every d, let $\mathbb{N}^n_d := \{\boldsymbol{\alpha} \in \mathbb{N}^n : |\boldsymbol{\alpha}| \leqslant d\}$, where $|\boldsymbol{\alpha}| = \sum_i \alpha_i$. In a multivariate setting, we denote $\boldsymbol{x}^{\boldsymbol{\beta}} = x_1^{\beta_1} \dots x_n^{\beta_n}$ and $\boldsymbol{\partial}^{\boldsymbol{\alpha}}_{\boldsymbol{x}} = \partial^{\alpha_1}_{x_1} \dots \partial^{\alpha_n}_{x_n}$ for $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{N}^n$. The derivative $\frac{\partial p}{\partial x_i}$ is denoted p_{x_i}. The indicator function of a set G is denoted by $\mathbb{1}_G$.

9.1.1 ▶ Selected problems

Let $G \subset \mathbb{R}^n$ be a bounded open set, whose boundary ∂G is algebraic (∂G is contained in the real zero set of finitely many polynomials), and let $\mu_f = f \mathbb{1}_G \mathrm{d}\boldsymbol{x}$ be a measure supported on G, with a so-called *holonomic weight* f against Lebesgue volume measure $\mathrm{d}\boldsymbol{x}$ on $\mathbb{R}^n$. This means that it satisfies a holonomic system (as defined in **Section 2.1.2**) as a generalized function if needed, see **Definition 9.8**. For instance, the weight $f(\boldsymbol{x}) = \exp(p(\boldsymbol{x}))$, with $p \in \mathbb{R}[\boldsymbol{x}]_s$ is holonomic i.e., it satisfies:

$$\left\{ \frac{\partial f}{\partial x_i} - \frac{\partial p}{\partial x_i} f = 0, \quad i \in [\![1, n]\!]. \right.$$

Consider also the power moments of μ_f:

$$m_{\boldsymbol{\alpha}} := \int_G \boldsymbol{x}^{\boldsymbol{\alpha}} \mathrm{d}\mu_f(\boldsymbol{x}), \quad \boldsymbol{\alpha} \in \mathbb{N}^n. \tag{9.1}$$

In [150], the following property is proved, for such an *exponential-polynomial* weight: knowing a priori the coefficients of p, its degree s and the degree d of the variety containing ∂G, a threshold N is identified (which depends only on d and s), such that the moments $m_{\boldsymbol{\alpha}}$ up to degree N (i.e. $\boldsymbol{\alpha} \in \mathbb{N}^n_N$) determine in a constructive and robust manner the coefficients of a polynomial vanishing on ∂G.

A natural question is whether this result can be generalized, as mentioned in [150]: *the analogy to the well understood moment rigidity of the Gaussian distribution is striking, although the constructive aspects of this finite determinateness remain too theoretical in general.* Motivated by this remark, in this chapter we revisit and extend this study to related problems, by exploiting holonomicity. In this framework, a first generalization of [150] is to recover the coefficients of both g and p in the *exponential-polynomial* case:

Problem 9.1 (Exp-Poly Inverse Problem) *Let $\mu_f = f\mathbb{1}_G d\boldsymbol{x}$ be a measure supported on a compact semi-algebraic set G, whose algebraic boundary is included in the zero set of a polynomial $g \in \mathbb{K}[\boldsymbol{x}]_d$. Let $f = \exp(p)$, with $p \in \mathbb{K}[\boldsymbol{x}]_s$. Given s, d, and a finite number of moments $m_{\boldsymbol{\alpha}}$, $|\boldsymbol{\alpha}| \leqslant N$, recover the coefficients of both g and p.*

More generally, the inverse problem for holonomic weights is:

Problem 9.2 (General Inverse Problem) *Let $\mu_f = f\mathbb{1}_G d\boldsymbol{x}$ be a measure supported on a compact semi-algebraic set G, with* holonomic *f. Given a finite number of moments $m_{\boldsymbol{\alpha}}$, $|\boldsymbol{\alpha}| \leqslant N$, recover a polynomial $g \in \mathbb{K}[\boldsymbol{x}]$ vanishing on the algebraic boundary of G and the coefficients of a holonomic system satisfied by f.*

Finally, we note the closely related direct problem:

Problem 9.3 (General Direct Problem) *Let $\mu_f = f\mathbb{1}_G d\boldsymbol{x}$ be a measure supported on a compact semi-algebraic set, with given* holonomic *f. Find a holonomic system of recurrences for the sequence of moments $(m_{\boldsymbol{\alpha}})$.*

Contributions. We address the above problems in the framework of holonomic distributions, employing well-known algorithmic properties of non-commutative polynomial representation of linear differential operators (see **Section 9.2**), as well as a generalized Stokes formula [150]. Firstly, this allows us to solve **Problem 9.1** in **Section 9.4.1**: we prove that this reconstruction problem boils down to solving a linear system of $3d + s - 1$ equations, involving moments up to degree $|\boldsymbol{\alpha}| \leqslant 4d + 2(s-1)$.

Secondly, as a by-product, an alternate method to creative telescoping is proposed for computing linear recurrences for the moments in **Section 9.3**. The advantage is that when the coefficients of g are given as parameters, the obtained recurrences stay linear with respect to them. However, there is no guarantee that this method provides a holonomic ideal. We could only prove that it solves **Problem 9.3** (i.e. it provides a holonomic ideal) in the restricted case of exponential-polynomial density and g nonsingular in $\mathbb{C}^n$.

Finally, **Problem 9.2** is solved in **Section 9.4.2**: we prove that a holonomic system for f can be found by solving a finite system of linear equations, but their number cannot be a priori bounded. Once the density is known, the support is reconstructed as solution of a similar linear system, but in this case we provide an explicit uniform bound on the number of required moments.

9.1.2 ▶ Related works

Moment problem. Concerning the moment problem, let μ be a Borel measure on $\mathbb{R}^n$ with all its moments finite. When μ is atomic with finitely many atoms (i.e., when $\mu = \sum_{k=1}^{d} \gamma_k \, \delta_{\boldsymbol{\xi_k}}$,

where $\delta_{\boldsymbol{\xi_k}}$ is the Dirac measure, for some $(\boldsymbol{\xi_k}) \subset \mathbb{R}^n$ and some positive weights (γ_k)), a first classical problem is to retrieve the atoms and the weights of μ from some finite truncation of its moment vector $(m_{\boldsymbol{\alpha}})_{\boldsymbol{\alpha}\in\mathbb{N}^n}$. In [175] a thorough overview of algebraic methods for this problem is given. An important idea consists in computing a sparse polynomial-exponential representation of a multivariate series from its truncated Taylor series, whose coefficients correspond to moments. For instance, for pairwise distinct $\boldsymbol{\xi_1}, \ldots, \boldsymbol{\xi_d}$, the moment generating series is:

$$\sigma_{\delta_{\boldsymbol{\xi}}}(\boldsymbol{y}) = \sum m_{\boldsymbol{\alpha}} \frac{\boldsymbol{y}^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!} = \sum_{k=1}^{d} \gamma_k \exp(\boldsymbol{\xi_k}^T \boldsymbol{y}).$$

Such generating functions are also the solutions of systems of partial differential equations with constant coefficients. Hence, the sparse representation of the polynomial-exponential (also known as Prony method) is related to the inverse system of the isolated points of the characteristic variety of this system. Methods to obtain such representation are given in [175]. Also, *flat extension* criteria, like for instance [148, Theorem 3.7], provide purely algebraic methods to reconstruct both the number of atoms, their values and weights function of the rank of the moment matrix.

All in all, moments of atomic measures satisfy multi-index linear recurrences with constant coefficients [175], which provide another incentive to consider the more general holonomic case. In this sense, these recurrences can be computed by creative telescoping.

Creative telescoping. These methods, briefly presented in **Chapter 2**, perform integration of functions (with free parameters), in the framework of non-commutative polynomial representation of linear differential operators (see [59, 141, 32] and references therein). In particular, the direct **Problem 9.3** can be solved for instance by the algorithms of Oaku [188]. Based on the D-module theory (see also [85, 240]), one computes a holonomic system for the definite integral of a holonomic function with parameters over a domain defined by polynomial inequalities. In the algorithms, holonomic distributions are involved, so, a subtle distinction has to be made between the ideal of operators with polynomial coefficients, which correspond to holonomicity, and those with rational coefficients which correspond to so-called D-finiteness (see **Section 2.1.2**).

Also, the Lagrange identity [122] (see also Equation (9.9) and **Proposition 9.13**), related to integration by parts, will play an important role in our approach. In the one variable case, for a linear differential operator with polynomial coefficients, $L = c_r\partial_x^r + \ldots + c_0$, its adjoint is defined as $L^* = (-1)^r\partial_x^r c_r + \ldots + c_0$ and the following holds:

$$\varphi L(f) - L^*(\varphi)f = \partial_x(\mathcal{L}_L(f,\varphi)), \tag{9.2}$$

for any function φ and f, with an explicit $\mathcal{L}_L$.

Inverse problem in the univariate case. In [15], the inverse **Problem 9.2** is solved in the univariate case, for piecewise D-finite densities. Specifically, $\mu_f = \sum_{i=1}^{d-1} \mathbb{1}_{[\xi_i,\xi_{i+1}]} f_i \mathrm{d}x$, for a set of d unknown points, $a = \xi_1 < \ldots < \xi_d = b$, with $[a,b] \subset \mathbb{R}$, and unknown smooth D-finite functions f_i. An important observation [15, Thm 2.12] is that the associated distribution $\sum_{i=1}^{d-1} \mathbb{1}_{[\xi_i,\xi_{i+1}]} f_i$ is annihilated by some holonomic operator $\hat{L} = g(x)^r L$, where $g(x) = \prod_{i=1}^{d}(x-\xi_i)$ and the operator L of order r satisfies $L \cdot f_i = 0$.

Remark 9.4 *As noted in* [15]*, for general holonomic operators L with $r > 1$, to correctly recover the parameters, the number N of required moments depends also on specific coefficients of L. An example is the nth Legendre polynomial, whose first n moments (taken over $[-1, 1]$) vanish, while $L_n = \partial_x((1-x^2)\partial_x) + n(n+1)$, hence the reconstruction of μ_f depends also on n, which enters the definition of L_n. On the contrary, for the* exponential-polynomial *case, we show that N depends only on the degrees of the polynomials involved.*

As discussed above, in the univariate case, the above problems are well tackled in literature, so this chapter deals with the multivariate case. However, to illustrate the basic ideas, we give two elementary univariate examples of our approach, omitting the technical proofs.

9.1.3 ▶ Introductory examples

Example 9.5 (Direct problem for erf-like function) *We are interested in computing a recurrence for the moments $m_i = \int_{-1}^{1} x^i e^{-x^2} \mathrm{d}x$. The idea is to include $\mathbb{1}_{[-1,1]}$ in the integral, and consider the distribution u corresponding to $\mathbb{1}_{[-1,1]}(x)e^{-x^2}$. Although not differentiable as a function, u satisfies (see* ***Section 9.2.1*** *for details):*

$$(1-x^2)(\partial_x + 2x)u = 0.$$

Integrating for the test function x^i, using (9.2) and noticing that its right hand side vanishes after integration, one has:

$$\int_{-1}^{1} e^{-x^2}(\partial_x + 2x)^*((1-x^2)x^i) = 0,$$

which directly provides the recurrence

$$im_{i-1} - (i+4)m_{i+1} + 2m_{i+3} = 0.$$

The extension of this method to the multivariate case is given in ***Section 9.3****.*

Example 9.6 (Univariate support and density reconstruction) *Consider the problem of reconstructing the parameters ξ_1, ξ_2 and p_2, p_1, p_0, provided the first N moments $\{m_i, 0 \leqslant i \leqslant N\}$ are known:*

$$m_i = \int_{\xi_1}^{\xi_2} x^i e^{p_2x^2+p_1x+p_0} \mathrm{d}x. \tag{9.3}$$

Like in the previous example, $u = \mathbb{1}_{[\xi_1,\xi_2]} e^{p_2x^2+p_1x+p_0}$ satisfies:

$$(x-\xi_1)(x-\xi_2)(\partial_x - 2p_2x - p_1)u = 0.$$

Denote by $\hat{L} := g(x)\partial_x + h(x)$ the operator to be reconstructed such that $\hat{L} \cdot f = 0$, with $g(x) = x^2 + g_1x + g_0$ and $h(x) = \sum_{i=0}^{3} h_i x^i$. Integrating and using the Lagrange identity, one has:

$$\int_{-\infty}^{\infty} \big(g(x)\partial_x - h(x))(x^i)\big)\, u \mathrm{d}x = 0.$$

This gives for each $i \geqslant 0$:

$$im_{i+1} + ig_1 m_i + ig_0 m_{i-1} - h_3 m_{i+3} - h_2 m_{i+2} - h_1 m_{i+1} - h_0 m_i = 0. \tag{9.4}$$

Hence, the coefficients of g and h are solutions of the above infinite linear system. If g is recovered, p (except for the coefficient p_0) could also be recovered from the division h/g. Finally p_0 can also be recovered from the equation (9.3), with $i = 0$.

The main question is whether a truncated system (9.4), which considers only moments up to degree N, can provide the correct solution for g and h. We will address this in ***Section 9.4****. Specifically, in* ***Theorem 9.18*** *we prove a sufficient bound for the case of an n-variable* exponential-polynomial *density, together with* ***Algorithm*** ReconstructExpPoly *which reconstructs the coefficients. It needs in our case the first $N = 10$ moments.*

9.2 Holonomic distributions and their moments

This section introduces the notion of *holonomic distributions*, which generalizes the notion of holonomicity defined in **Chapter 2** to distributions, seen as *generalized functions*. In this particular setting, distinguishing *D-finiteness* and *holonomicity* is crucial. We refer to **[59, 141, 188]** for a more comprehensive presentation.

9.2.1 ▶ Holonomic distributions

Introduced by Schwartz **[225]**, *distributions* generalize functions and measures. A minimal introduction to this topic is provided below.

Definition 9.7 (Test functions and distributions) *Let $\mathcal{E} = \mathcal{C}^\infty(\mathbb{R}^n)$ be the set of smooth functions over $\mathbb{R}^n$, equipped with the* compact-open topology*: $\varphi_k \to \varphi$ in $\mathcal{E}$ if $\partial_{\boldsymbol{x}}^{\boldsymbol{\alpha}} \cdot \varphi_k$ converges uniformly to $\partial_{\boldsymbol{x}}^{\boldsymbol{\alpha}} \cdot \varphi$ over every compact set, for each $\boldsymbol{\alpha} \in \mathbb{N}^n$.*

Its topological dual $\mathcal{E}'$ is the set of compactly supported distributions *(or simply* distributions *in this chapter) i.e. linear forms $T : \mathcal{E} \to \mathbb{R}$ such that:*

- *There exists a minimal compact set $K \subseteq \mathbb{R}^n$ (the* support *of T) such that $\langle T, \varphi \rangle = 0$ whenever φ vanishes over K.*
- *$\langle T, \varphi_k \rangle \to 0$ whenever $\varphi_k \to 0$ in $\mathcal{E}$.*

$\mathcal{E}'$ has a canonical $\mathfrak{D}_n$-module structure:

$$\langle L \cdot T, \varphi \rangle := \langle T, L^* \cdot \varphi \rangle, \qquad L \in \mathfrak{D}_n,\ T \in \mathcal{E}',\ \varphi \in \mathcal{E},$$

where the adjoint operator *L^* is defined by*

$$x_i^* = x_i, \qquad \partial_{x_i}^* = -\partial_{x_i}, \quad \text{and} \quad (L_1 L_2)^* = L_2^* L_1^*.$$

Definition 9.8 (Holonomic distribution) *A distribution $T \in \mathcal{E}'$ is holonomic if its annihilator is a holonomic ideal of $\mathfrak{D}_n$:*

$$\mathfrak{Ann}(T) := \{L \in \mathfrak{D}_n \mid L \cdot T = 0 \text{ as a distribution}\}.$$

A measure supported on a set G, with density $f \in \mathcal{E}$, is represented by the distribution $f\mathbb{1}_G$, with $\langle f\mathbb{1}_G, \varphi\rangle = \int_G \varphi(\boldsymbol{x})f(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$.

We make the following assumption on $G \subseteq \mathbb{R}^n$:

Assumption 9.9 *G is a compact n-dimensional semi-algebraic set. In particular, the following holds:*

(1) G is an n-dimensional compact manifold such that its boundary can be decomposed as $\partial G = Z \cup Z'$, with Z a finite union of $(n-1)$-dimensional manifolds and Z' a negligible set w.r.t. the $(n-1)$-dimensional Hausdorff measure.

(2) the ideal of polynomials vanishing over ∂G is radical and principal i.e., generated by a single square-free polynomial g. In particular, the family $\{g, g_{x_1}, \ldots, g_{x_n}\}$ is coprime, implying that the set of singular points $\{\boldsymbol{x} \mid g(\boldsymbol{x}) = 0 \text{ and } \nabla g(\boldsymbol{x}) = 0\}$ is negligible in ∂G.

9.2.2 ▶ Moments of a distribution

Definition 9.10 (Moments of a compactly supported distribution) *The moments of a distribution $T \in \mathcal{E}'$ are:*

$$m_{\boldsymbol{\alpha}}(T) := \langle T, \boldsymbol{x}^{\boldsymbol{\alpha}}\rangle, \qquad \boldsymbol{\alpha} \in \mathbb{N}^n. \tag{9.5}$$

Note that if $T = f\mathbb{1}_G$ with G compact and $f \in \mathcal{E}$, then $m_{\boldsymbol{\alpha}}(f\mathbb{1}_G)$ coincides with the moments defined in Equation (9.1). A convenient way to deal with moments of a distribution is the *Fourier transform* (also called *characteristic function*).

Definition 9.11 *The Fourier transform of a distribution $T \in \mathcal{E}'$ is the analytic function $\mathscr{F}\{T\}$ of $\boldsymbol{z} = (z_1, \ldots, z_n) \in \mathbb{R}^n$ defined by:*

$$\mathscr{F}\{T\}(\boldsymbol{z}) = \sum_{\boldsymbol{\alpha}\in\mathbb{N}^n} m_{\boldsymbol{\alpha}}(T)\frac{(-\mathrm{i}\,\boldsymbol{z})^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!} = \langle T, e^{-\mathrm{i}\,\boldsymbol{x}^T\boldsymbol{z}}\rangle, \qquad \boldsymbol{z} \in \mathbb{R}^n.$$

Proposition 9.12 *Let $T \in \mathcal{E}'$ and $L = \sum_{\boldsymbol{\beta}} q_{\boldsymbol{\beta}}(\boldsymbol{x})\boldsymbol{\partial}_{\boldsymbol{x}}^{\boldsymbol{\beta}} \in \mathfrak{D}_n$.*

(i) The Fourier transform of $L \cdot T$ is related to that of T by

$$\mathscr{F}\{L \cdot T\} = L^{\mathscr{F}} \cdot \mathscr{F}\{T\}, \quad \text{with} \quad L^{\mathscr{F}} := L\begin{bmatrix} x_i & \mapsto \mathrm{i}\,\partial_{z_i} \\ \partial_{x_i} & \mapsto \mathrm{i}\,z_i \end{bmatrix} = \sum_{\boldsymbol{\beta}} q_{\boldsymbol{\beta}}(\mathrm{i}\,\boldsymbol{\partial}_{\boldsymbol{z}})(\mathrm{i}\,\boldsymbol{z})^{\boldsymbol{\beta}}, \tag{9.6}$$

where i denotes the complex number of positive imaginary part satisfying $\mathrm{i}^2 = -1$.

(ii) The moments of $L \cdot T$ are related to those of T by

$$(m_{\boldsymbol{\alpha}}(L \cdot T)) = L^{\mathscr{M}} \cdot (m_{\boldsymbol{\alpha}}(T)), \qquad \text{with}$$
$$L^{\mathscr{M}} := L\begin{bmatrix} x_i & \mapsto S_{\alpha_i} \\ \partial_{x_i} & \mapsto -\alpha_i S_{\alpha_i}^{-1} \end{bmatrix} = \sum_{\boldsymbol{\beta}} (-1)^{|\boldsymbol{\beta}|} q_{\boldsymbol{\beta}}(\boldsymbol{S}_{\boldsymbol{\alpha}}) \left(\prod_{i=1}^{n} (\alpha_i S_{\alpha_i}^{-1})^{\beta_i}\right). \tag{9.7}$$

The proof is very similar to [175, Sec. 5.1.]. We give it here for the sake of completeness.

Proof of Proposition 9.12. To prove (i), we use $\partial_{x_i} e^{-\mathrm{i}\boldsymbol{x}^T\boldsymbol{z}} = -\mathrm{i}\, z_i e^{-\mathrm{i}\boldsymbol{x}^T\boldsymbol{z}}$ and $\partial_{z_i} e^{-\mathrm{i}\boldsymbol{x}^T\boldsymbol{z}} = -\mathrm{i}\, x_i e^{-\mathrm{i}\boldsymbol{x}^T\boldsymbol{z}}$:

$$\begin{aligned}
\mathscr{F}\{L\cdot T\} &= \langle T, L^*\cdot(e^{-\mathrm{i}\boldsymbol{x}^T\boldsymbol{z}})\rangle \\
&= \langle T, \sum_{\boldsymbol{\beta}} (-1)^{|\boldsymbol{\beta}|}\boldsymbol{\partial}_{\boldsymbol{x}}^{\boldsymbol{\beta}}(q_{\boldsymbol{\beta}}(\boldsymbol{x})\cdot e^{-\mathrm{i}\boldsymbol{x}^T\boldsymbol{z}})\rangle \qquad \text{by definition,} \\
&= \langle T, \sum_{\boldsymbol{\beta}} (-1)^{|\boldsymbol{\beta}|} q_{\boldsymbol{\beta}}(\mathrm{i}\,\boldsymbol{\partial}_{\boldsymbol{z}})\boldsymbol{\partial}_{\boldsymbol{x}}^{\boldsymbol{\beta}}\cdot(e^{-\mathrm{i}\boldsymbol{x}^T\boldsymbol{z}})\rangle \\
&= \langle T, \sum_{\boldsymbol{\beta}} (-1)^{|\boldsymbol{\beta}|} q_{\boldsymbol{\beta}}(\mathrm{i}\,\boldsymbol{\partial}_{\boldsymbol{z}})(-\mathrm{i}\,\boldsymbol{z})^{\boldsymbol{\beta}}\cdot e^{-\mathrm{i}\boldsymbol{x}^T\boldsymbol{z}}\rangle \\
&= \langle T, L^{\mathscr{F}}\cdot e^{-\mathrm{i}\boldsymbol{x}^T\boldsymbol{z}}\rangle \;=\; L^{\mathscr{F}}\cdot\mathscr{F}\{T\}.
\end{aligned}$$

The last equality holds since for any $\mathcal{C}^\infty$ function $f(x,z)$,

$$\begin{aligned}
\langle T, z_i\cdot f(\boldsymbol{x},\boldsymbol{z})\rangle &= z_i\cdot\langle T, f(\boldsymbol{x},\boldsymbol{z})\rangle, \qquad\qquad \text{and} \\
\langle T, \partial_{z_i}\cdot f(\boldsymbol{x},\boldsymbol{z})\rangle &= \left\langle T, \lim_{h\to 0}\frac{f(\boldsymbol{x},\boldsymbol{z}+h\boldsymbol{e_i})-f(\boldsymbol{x},\boldsymbol{z})}{h}\right\rangle \\
&= \lim_{h\to 0}\frac{1}{h}\left(\langle T, f(\boldsymbol{x},\boldsymbol{z}+h\boldsymbol{e_i})\rangle - \langle T, f(\boldsymbol{x},\boldsymbol{z})\rangle\right) \\
&= \partial_{z_i}\cdot\langle T, f(\boldsymbol{x},\boldsymbol{z})\rangle,
\end{aligned}$$

where the commutation of the limit symbol comes from the fact that $f_{\boldsymbol{z},h} : \boldsymbol{x}\mapsto \frac{f(\boldsymbol{x},\boldsymbol{z}+h\boldsymbol{e_i})-f(\boldsymbol{x},\boldsymbol{z})}{h}$ converges to $f_{\boldsymbol{z}} : \boldsymbol{x}\mapsto f(\boldsymbol{x},\boldsymbol{z})$ for the compact-open topology of $\mathcal{E}$.

To prove (ii), one just need to notice that $L^{\mathscr{M}}$ is obtained from $L^{\mathscr{F}}$ using

$$z_i \mapsto \mathrm{i}\,\alpha_i S_{\alpha_i}^{-1}, \qquad \text{and} \qquad \partial_{z_i}\mapsto -\mathrm{i}\, S_{\alpha_i}.$$

□

■ **Proposition 9.13** *Let $T\in\mathcal{E}'$. An operator $L\in\mathfrak{D}_n$ satisfies*

$$\langle T, L^*\cdot\boldsymbol{x}^{\boldsymbol{\alpha}}\rangle = 0, \qquad \textit{for all } \boldsymbol{\alpha}\in\mathbb{N}^n, \tag{9.8}$$

if and only if $L\in\mathfrak{Ann}(T)$.

Proof. By the injectivity of the Fourier transform on compactly supported distributions [225]. □

9.3 Direct problem for moments

As mentioned in the introduction, the direct **Problem 9.3** can be solved using an algorithm presented in [188]. However, one may ask whether the simple roadmap of **Example 9.5** can be generalized to the multivariate case and provide a more efficient method. For that, firstly, Lagrange identity in the multivariate setting is needed:

Lemma 9.14 (Lagrange identity) *For $f, g \in \mathcal{E}$ and $L \in \mathfrak{D}_n$ of order r, there exists a vector field $\mathcal{L}_L(f,g) : \mathbb{R}^n \to \mathbb{R}^n$, called bilinear concomitant, depending on L and linear in f and g, such that:*

$$(L \cdot f)\, g - f\, (L^* \cdot g) = \nabla \cdot \mathcal{L}_L(f,g). \tag{9.9}$$

Each component of $\mathcal{L}_L(f,g)$ can be written

$$\mathcal{L}_{L,i}(f,g) = \sum_{|\boldsymbol{\alpha}|+|\boldsymbol{\beta}| \leqslant r-1} c_{L,i,\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x})(\boldsymbol{\partial}_{\boldsymbol{x}}^{\boldsymbol{\alpha}} \cdot f)(\boldsymbol{\partial}_{\boldsymbol{x}}^{\boldsymbol{\beta}} \cdot g), \qquad i \in [\![1,n]\!].$$

with coefficients $c_{L,i,\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{x}) \in \mathbb{K}[\boldsymbol{x}]$ depending on L.

Secondly, the action of differential operators on compactly supported distributions of the form $f\mathbb{1}_G$ is provided:

Proposition 9.15 *Let G be as in **Assumption 9.9**, $f \in \mathcal{E}$ and $L \in \mathfrak{D}_n$. Then the distribution $L \cdot (f\mathbb{1}_G)$ admits the following expression:*

$$\langle L \cdot (f\mathbb{1}_G), \varphi \rangle = \int_G \varphi\, (L \cdot f) \mathrm{d}x - \int_{\partial G} \mathcal{L}_L(f,\varphi) \cdot \boldsymbol{n}\ \mathrm{d}S,$$

where $\boldsymbol{n}$ and $\mathrm{d}S$ respectively denote the normal vector and the $(n-1)$-dimensional Hausdorff measure on ∂G.

Proof. Integrating Lagrange's identity (9.9) with $g = \varphi$ and using the divergence theorem, we have:

$$\int_G \varphi\, (L \cdot f) \mathrm{d}\boldsymbol{x} - \int_G (L^* \cdot \varphi)\, f \mathrm{d}\boldsymbol{x} = \int_{\partial G} \mathcal{L}_L(f,\varphi) \cdot \boldsymbol{n}\ \mathrm{d}S.$$

Following [150], the divergence theorem is a consequence of Stokes' theorem when ∂G is smooth, or of a generalization by Whitney [261, Theorem 14A] when G satisfies **Assumption 9.9**.(1). □

Finally, the following proposition provides differential equations for measures supported on semi-algebraic sets.

Proposition 9.16 *Let G and g be as in **Assumption 9.9**, $f \in \mathcal{E}$, $L \in \mathfrak{Ann}(f)$ of order r. Then $g^r L \in \mathfrak{Ann}(f\mathbb{1}_G)$.*

Proof. Using **Proposition 9.15**, one needs to prove that $\int_G g^r \varphi (L \cdot f) \mathrm{d}\boldsymbol{x}$ and $\int_{\partial G} \mathcal{L}_L(f, g^r\varphi) \cdot \boldsymbol{n}\ \mathrm{d}S$ are zero. The first one is trivial since $L \in \mathfrak{Ann}(f)$. For the second, $\mathcal{L}_L(f, g^r\varphi)$ involves derivatives $\boldsymbol{\partial}_{\boldsymbol{x}}^{\boldsymbol{\alpha}}(g^r\varphi)$ with $|\boldsymbol{\alpha}| < r$ (**Lemma 9.14**), so it vanishes over ∂G. □

Hence, **Proposition 9.16** gives an easy way to construct operators in $\mathfrak{Ann}(f\mathbb{1}_G)$ from operators in $\mathfrak{Ann}(f)$. Indeed, given a Gröbner basis $\{L_1, \ldots, L_k\}$ of $\mathfrak{Ann}(f)$, and $g \in \mathbb{R}[\boldsymbol{x}]$ vanishing over ∂G, each operator $g^{r_i} L_i$ (with r_i the order of L_i) annihilates $f\mathbb{1}_G$ as a distribution. Therefore, each operator $R_i := (g^{r_i} L_i)^{\mathscr{M}}$ gives a valid recurrence for the sequence of moments $(m_{\boldsymbol{\alpha}})$ (see equation (9.7)).

However, from the fact that f is holonomic one can not directly guarantee that the ideal generated by $\{g^{r_1} L_1, \ldots, g^{r_k} L_k\}$ is holonomic. Similarly, we are not able to prove (or refute) that $\{R_1, \ldots, R_k\}$ is holonomic in general. Nevertheless, one can apply a Gröbner basis algorithm, which will possibly return a basis of a holonomic ideal. This heuristic is given in **Algorithm** RecurrencesMoments. We prove that this algorithm actually returns such a basis in the particular case of an exponential-polynomial density (including the Lebesgue measure), and a smooth boundary, extending [188, Prop. 4].

Algorithm 9.1 RecurrencesMoments$(n, g, \{L_1, \dots, L_k\})$

Input: Gröbner basis $\{L_1, \dots, L_k\}$ for $\mathfrak{Ann}(f)$, g.
Output: Gröbner basis for $\mathfrak{Ann}(m_{\boldsymbol{\alpha}})$.

1: $R_i \leftarrow (g^{r_i} L_i)^{\mathscr{M}}$, as in (9.7), with r_i the order of L_i, for $i \in [\![1, k]\!]$
2: **return** GröbnerBasis($\{R_1, \dots, R_k\}$,$\mathfrak{R}_n$)

Proposition 9.17 *Let $f(\boldsymbol{x}) = e^{p(\boldsymbol{x})}$ with $p \in \mathbb{R}[\boldsymbol{x}]_s$, and $g \in \mathbb{R}[\boldsymbol{x}]_d$ vanishing over ∂G. Suppose moreover that g is nonsingular in $\mathbb{C}^n$, that is, there exists no $\boldsymbol{x} \in \mathbb{C}^n$ such that $g(\boldsymbol{x}) = 0$ and $\nabla g(\boldsymbol{x}) = 0$.*

(i) The operators

$$L_i = g(\partial_{x_i} - p_{x_i}), \qquad i \in [\![1, n]\!],$$

are generators of a holonomic ideal $\mathfrak{I}$ contained in $\mathfrak{Ann}(f\mathbb{1}_G)$.

(ii) The operators $L_i^{\mathscr{F}}$ $(i \in [\![1, n]\!])$ span a holonomic ideal $\mathfrak{I}^{\mathscr{F}}$ contained in $\mathfrak{Ann}(\mathscr{F}\{f\mathbb{1}_G\})$.

Proof. For (i), first note that the operators L_i also generate

$$\begin{aligned} L_{ij} &:= (\partial_{x_j} - p_{x_j})L_i - (\partial_{x_i} - p_{x_i})L_j \\ &= g_{x_j}(\partial_{x_i} - p_{x_i}) - g_{x_i}(\partial_{x_j} - p_{x_j}), \qquad 1 \leqslant i < j \leqslant n. \end{aligned} \tag{9.10}$$

Holonomicity is proved via the *characteristic variety*, as for instance in [188]. For $L = \sum_{|\boldsymbol{\alpha}| \leqslant r} q_{\boldsymbol{\alpha}}(x)\boldsymbol{\partial}_{\boldsymbol{x}}^{\boldsymbol{\alpha}}$ of order r, define its principal symbol $\sigma(L)(\boldsymbol{x}, \boldsymbol{\xi}) = \sum_{|\boldsymbol{\alpha}| = r} q_{\boldsymbol{\alpha}}(\boldsymbol{x})\boldsymbol{\xi}^{\boldsymbol{\alpha}}$ for $(\boldsymbol{x}, \boldsymbol{\xi}) \in \mathbb{C}^{2n}$. Then for a left ideal $\mathfrak{I}$, $\mathrm{Char}(\mathfrak{I}) = \{(\boldsymbol{x}, \boldsymbol{\xi}) \in \mathbb{C}^{2n} \mid \sigma(L)(\boldsymbol{x}, \boldsymbol{\xi}) = 0 \ \forall L \in \mathfrak{I} \setminus \{0\}\}$. With these notations, $\mathfrak{I}$ is holonomic if and only if all the components of $\mathrm{Char}(\mathfrak{I})$ are of dimension at most n. In our case,

$$\sigma(L_i)(\boldsymbol{x}, \boldsymbol{\xi}) = g(\boldsymbol{x})\xi_i, \quad \text{and} \quad \sigma(L_{ij})(\boldsymbol{x}, \boldsymbol{\xi}) = g_{x_j}(\boldsymbol{x})\xi_i - g_{x_i}(\boldsymbol{x})\xi_j.$$

Hence, if $(\boldsymbol{x}, \boldsymbol{\xi}) \in \mathrm{Char}(\mathfrak{I})$, then either $g(\boldsymbol{x}) \neq 0$, implying $\boldsymbol{\xi} = 0$, or $g(\boldsymbol{x}) = 0$. In the latter case, $\nabla g(\boldsymbol{x}) \neq 0$ (since g is nonsingular) and hence there exists $\lambda \in \mathbb{C}$ s.t. $\boldsymbol{\xi} = \lambda \nabla g(\boldsymbol{x})$. In both cases, the corresponding components of $\mathrm{Char}(\mathfrak{I})$ have dimension n.

For (ii), since the Fourier transform maps x_i to $\mathrm{i}\,\partial_{z_i}$ and ∂_{x_i} to $\mathrm{i}\,z_i$, it is clear that $\mathfrak{I}$ is holonomic if and only if $\mathfrak{I}^{\mathscr{F}}$ is holonomic. □

Interestingly enough, for the examples we tried for an exponential-polynomial density, **Algorithm** RecurrencesMoments always terminated, even when the boundary was not smooth (see **Example 9.24**). Also, it was faster than "classical" creative telescoping, which firstly constructs a Gröbner basis for $f\mathbb{1}_G$ and then applies Takayama's algorithm [188]. Further investigation is needed to provide a comparison in this case.

However, having a Gröbner basis is not mandatory for the reconstruction problem addressed in the next section. The recurrences obtained as above turn out to be sufficient and constitute the basic brick of our reconstruction method.

9.4 | Reconstruction methods

Given some moments $m_{\boldsymbol{\alpha}}(f\mathbb{1}_G)$ associated to a measure of unknown D-finite density $f \in \mathcal{E}$ and unknown compact algebraic support G, our goal is to reconstruct a polynomial $\tilde{g}$ vanishing on the boundary ∂G of G and operators $\tilde{L} \in \mathfrak{Ann}(f)$.

The general approach is the following:

- Take an *ansatz* $L' = \sum_{\boldsymbol{\beta}\in A} q_{\boldsymbol{\beta}}(\boldsymbol{x})\boldsymbol{\partial}_{\boldsymbol{x}}^{\boldsymbol{\beta}}$, for a specified finite set $A \subset \mathbb{N}^n$ and polynomials $q_{\boldsymbol{\beta}}(\boldsymbol{x})$ with specified degrees $d_{\boldsymbol{\beta}}$.
- Let $R = L'^{\mathscr{M}}$. Solve a finite-dimensional linear system in the unknown coefficients of the polynomials $q_{\boldsymbol{\beta}}$:
$$(R \cdot m(f\mathbb{1}_G))_{\boldsymbol{\alpha}} = 0, \qquad |\boldsymbol{\alpha}| \leqslant N. \tag{9.11}$$
This requires the knowledge of moments $m_{\boldsymbol{\alpha}}(f\mathbb{1}_G)$ with $|\boldsymbol{\alpha}| \leqslant N + \max_{\boldsymbol{\beta}\in A}\{d_{\boldsymbol{\beta}} - |\boldsymbol{\beta}|\}$ (see eq. (9.7)).
- From the solution L' of (9.11), extract a polynomial $\tilde{g}$ vanishing on ∂G and an operator $\tilde{L} \in \mathfrak{Ann}(f)$.

Note that the solution of (9.11) corresponds to a truncation of the infinite system (9.8), since $\langle f\mathbb{1}_G, L'^* \cdot \boldsymbol{x}^{\boldsymbol{\alpha}}\rangle = 0$, for $|\boldsymbol{\alpha}| \leqslant N$. Hence one is interested in obtaining bounds $\hat{N}$ on N, such that any solution of (9.11) is also solution of (9.8). Such an *a priori* uniform bound depending only on A and $d_{\boldsymbol{\beta}}$ does not exist in general, cf. **Remark 9.4**.

Another issue is that L' may not be factorized as $\tilde{g}(\boldsymbol{x})^r\tilde{L}$ with $\tilde{g}$ vanishing on ∂G and $\tilde{L}\cdot f = 0$. See for instance the operator in (9.10).

In **Section 9.4.1**, we solve both issues when f is exponential-polynomial and give the associated algorithm. Then, in **Section 9.4.2**, we address the general holonomic case in two steps: firstly, for recovering the density, we prove that N is finite, but no *a priori* bound for it is known; secondly, once the density is known, a stronger result is proved for the support reconstruction, since an explicit uniform bound on the number of required moments is given.

In what follows, "exact computations" are assumed, that is, both the polynomial coefficients and the given moments $m_{\boldsymbol{\alpha}}$ lie in a computable finite extension of $\mathbb{Q}$. The practical case of approximately known numerical moments is briefly analyzed in **Section 9.5**.

9.4.1 ▶ Exponential-polynomial densities

Let $f(\boldsymbol{x}) = \exp(p(\boldsymbol{x}))$ with $\deg p = s$, together with G and g be as in **Assumption 9.9**, $\deg g = d$. Then f is annihilated by $L_i = \partial_{x_i} - p_{x_i}$ for $i \in [\![1, n]\!]$. **Algorithm** ReconstructExpPoly follows the general approach above, with ansatz $L'_i = h_0\partial_{x_i} - h_i$ $(i \in [\![1, n]\!])$ for unknown polynomials $h_0, \ldots, h_n$ where $\deg h_0 \leqslant d$ and $\deg h_i \leqslant d + s - 1$ for $i \in [\![1, n]\!]$. **Theorem 9.18** establishes its correctness, with an explicit bound $\hat{N}$.

Algorithm 9.2 ReconstructExpPoly$(n, d, s, N, (m_{\boldsymbol{\alpha}})_{|\boldsymbol{\alpha}|\leqslant N+d+s-1})$

Input: $n \geqslant 2$, degrees $d, s \geqslant 0$, moments $m_{\boldsymbol{\alpha}}$ for $|\boldsymbol{\alpha}| \leqslant N+d+s-1$.
Output: $\tilde{g}, \tilde{p} \in \mathbb{K}[\boldsymbol{x}]$ with $\deg(\tilde{g}) \leqslant d$ and $\deg(\tilde{p}) \leqslant s$.

▷ *Find* $L_i' \in \mathfrak{Ann}(f\mathbb{1}_G)$
1: $h_0 \leftarrow \sum_{|\boldsymbol{\gamma}|\leqslant d} h_{0\boldsymbol{\gamma}}\boldsymbol{x}^{\boldsymbol{\gamma}}$ and $h_i \leftarrow \sum_{|\boldsymbol{\gamma}|\leqslant d+s-1} h_{i\boldsymbol{\gamma}}\boldsymbol{x}^{\boldsymbol{\gamma}}$ for $i \in [\![1, n]\!]$, with symbolic coefficients $h_{i\boldsymbol{\gamma}}$
2: $L_i' \leftarrow h_0\partial_{x_i} - h_i$ for $i \in [\![1, n]\!]$
3: Find a nontrivial solution $\{h_{i\boldsymbol{\gamma}}\}$ of the linear system:

$$(L_i'^{\mathscr{M}} m)_{\boldsymbol{\alpha}} = 0, \qquad i \in [\![1, n]\!],\ |\boldsymbol{\alpha}| \leqslant N$$

▷ *Reconstruct* $\tilde{g}$ *and* $\tilde{p}$
4: $\tilde{g} \leftarrow h_0$ and $\tilde{p}_i \leftarrow h_i/\tilde{g}$ for $i \in [\![1, n]\!]$
5: $\tilde{p} \leftarrow \sum_{i=1}^{n} \int_0^{x_i} \tilde{p}_i(0, \dots, 0, t_i, x_{i+1}, \dots, x_n)\mathrm{d}t_i$
6: **return** $(\tilde{g}, \tilde{p})$

■ **Theorem 9.18** *Let* $f(\boldsymbol{x}) = \exp(p(\boldsymbol{x}))$ *with* $\deg p = s$, *and* G, g *with* $\deg g = d$ *be as in* ***Assumption 9.9****. If* $N \geqslant \hat{N} = 3d + s - 1$, *then* ReconstructExpPoly$(n, d, s, N, (m_\alpha))$ *returns* $\tilde{g} = \lambda g$ *with* $\lambda \in \mathbb{K}^*$, *and* $\tilde{p} = p - p(0)$. *This requires moments up to degree* $4d+2(s-1)$.

Moreover, if $g \geqslant 0$ *over* G, $\hat{N}$ *can be only* $2d + s - 1$, *requiring moments up to degree* $3d + 2(s - 1)$.

■ **Remark 9.19** *This method cannot reconstruct the constant coefficient of* p, *which is the scaling factor of the density. In case of a probability measure over* $\mathbb{R}^n$, *this coefficient is uniquely recovered by imposing* $\int_{\mathbb{R}^n} \exp(\tilde{p}(\boldsymbol{x}))\mathrm{d}\boldsymbol{x} = 1$. *Otherwise, one can compute* $p(0) = \log(m_0/\int_G \exp(\tilde{p}(\boldsymbol{x}))\mathrm{d}\boldsymbol{x})$, *for example.*

Proof. First, $\{h_0 \leftarrow g, h_i \leftarrow gp_{x_i}, i \in [\![1, n]\!]\}$ is a solution of the linear system in line 3. Hence, one can always get a solution with $h_0 \neq 0$. Then $\langle L_i'(f\mathbb{1}_G), \varphi\rangle = 0$ for all $i \in [\![1, n]\!]$ and $\varphi \in \mathbb{K}[\boldsymbol{x}]_N$. Using **Proposition 9.15**, this expands to:

$$\int_G (h_0 p_{x_i} - h_i)\varphi f \mathrm{d}\boldsymbol{x} + \int_{\partial G} h_0 \varphi f\ \boldsymbol{e_i} \cdot \boldsymbol{n}\ \mathrm{d}S = 0. \tag{9.12}$$

With $\varphi = (h_0 p_{x_i} - h_i)g^2$ of degree at most $d + (s - 1) + 2d \leqslant N$, the second integral is zero since g vanishes over ∂G. Hence the first integral is zero too. Therefore, its integrand $(h_0 p_{x_i} - h_i)^2 g^2$ is zero almost everywhere over G. Since G has nonempty interior and $f > 0$, $g \neq 0$, this necessarily implies $h_i = h_0 p_{x_i}$ for all $i \in [\![1, n]\!]$.

Now, the first integral in (9.12) being always zero for all polynomials φ with $\deg\varphi \leqslant N$, so is the second. Noticing that $\boldsymbol{e_i} \cdot \boldsymbol{n} = g_{x_i}/\|\nabla g\|$ when $\nabla g \neq 0$, and by taking $\varphi = h_0 g_{x_i}$ of degree at most $2d - 1 \leqslant N$, we have

$$\int_{\partial G} (h_0 g_{x_i})^2 \frac{f}{\|\nabla g\|}\mathrm{d}S = 0.$$

By summing this equality for $i \in [\![1, n]\!]$, we get that $h_0\|\nabla g\|$ vanishes over ∂G. Since by **Assumption 9.9**.(2), $\{\boldsymbol{x} \in \partial G \mid \nabla g(\boldsymbol{x}) = 0\}$ is negligible in ∂G, we have that h_0 (of degree at most d) vanishes over ∂G, whence $h_0 = \lambda g$ since g is square-free.

Finally, $\tilde{p} = p - p(0)$ is reconstructed from $p_{x_i} = \tilde{p}_i$ in line 5.

For the case where $g \geqslant 0$ over ∂G, the first step of the proof still holds with $\varphi = (h_0 p_{x_i} - h_i)g$, of degree $2d + s - 1$, in **(9.12)**. □

9.4.2 ▶ Holonomic densities

For higher order holonomic operators, the proof of **Theorem 9.18** cannot be generalized: the key argument for deducing a uniform bound $\hat{N}$ was to write in **(9.12)**, $\int_G \varphi(L'f)\mathrm{d}\boldsymbol{x}$ as $\int_G h\varphi f \mathrm{d}\boldsymbol{x}$, with $h \in \mathbb{K}[\boldsymbol{x}]$.

Instead, we proceed in two steps. Firstly in **Section 9.4.2**, a holonomic system for f is reconstructed, but it requires a finite number N of linear equations, which cannot be *a priori* bounded. Secondly, the support is reconstructed in **Section 9.4.2**.

Reconstructing the density

Algorithm ReconstructDensity produces a holonomic ideal $\mathfrak{I} \subseteq \mathfrak{Ann}(f)$ spanned by a *rectangular system* $\{L_1, \ldots, L_n\}$, that is $L_i \in \mathfrak{Ann}(f) \cap \mathbb{K}[\boldsymbol{x}]\langle\partial_{x_i}\rangle$ only involves derivatives w.r.t. x_i. For that, it is sufficient to find operators annihilating $f\mathbb{1}_G$.

Proposition 9.20 *Let f be analytic over G satisfying **Assumption 9.9**. Then $\mathfrak{Ann}(f\mathbb{1}_G) \subseteq \mathfrak{Ann}(f)$.*

Proof. Let $L' \in \mathfrak{Ann}(f\mathbb{1}_G)$ be of order r. **Proposition 9.15** with $\varphi = g^{2r}(L' \cdot f)$ gives:

$$\int_G g^{2r}(L' \cdot f)^2 \mathrm{d}\boldsymbol{x} = 0.$$

This implies that the analytic function $g^r(L' \cdot f)$ vanishes over G of nonempty interior, hence is 0. Since $g \neq 0$, $L' \cdot f = 0$. □

Theorem 9.21 guarantees that **Algorithm** ReconstructDensity always returns an $L \in \mathfrak{Ann}(f)$ for N large enough.

Theorem 9.21 *Let $i \in [\![1, n]\!]$, f be analytic, $G, g \in \mathbb{K}[\boldsymbol{x}]_d$ satisfying **Assumption 9.9**, and let $L = \sum_{j=0}^{r} q_j(\boldsymbol{x})\partial_{x_i}^j \in \mathfrak{Ann}(f) \cap \mathbb{K}[\boldsymbol{x}]\langle\partial_{x_i}\rangle$ be of minimal order r, with q_r of minimal degree. Then, **Algorithm** ReconstructDensity$(n, i, r, s, N, (m_{\boldsymbol{\alpha}}))$ returns $\tilde{L} = \lambda L$ with $\lambda \in \mathbb{K}^*$ for $s \geqslant dr + \max\{\deg(q_j)\}$ and N large enough.*

Proof. The linear system in line 3 always has $g^r L$ as solution, by **Proposition 9.16**. Now let K_N denote the kernel of this system, that is $L' \in K_N$ if and only if $\langle L'(f\mathbb{1}_G), \boldsymbol{x}^{\boldsymbol{\alpha}}\rangle = 0$ for all $|\boldsymbol{\alpha}| \leqslant N$. The infinite inclusion chain of finite-dimensional linear subspaces $\cdots \supseteq K_N \supseteq K_{N+1} \supseteq \ldots$ is necessarily stationary. So for N large enough, $L' \in K_N$ implies $\langle L' \cdot (f\mathbb{1}_G), \boldsymbol{x}^{\boldsymbol{\alpha}}\rangle = 0$ for all $\boldsymbol{\alpha}$ and hence $L' \in \mathfrak{Ann}(f\mathbb{1}_G)$ by **Proposition 9.13**. Finally, $L' \in \mathfrak{Ann}(f)$ by **Proposition 9.20**

Algorithm 9.3 ReconstructDensity$(n, i, r, s, N, (m_{\boldsymbol{\alpha}})_{|\boldsymbol{\alpha}|\leqslant N+s})$

Input: $n \geqslant 2$, $i \in [\![1, n]\!]$, order r, maximum degree s, moments $m_{\boldsymbol{\alpha}}$ for $|\boldsymbol{\alpha}| \leqslant N + s$.

Output: $\tilde{L} = \sum_{j=0}^{r} \tilde{q}_j(\boldsymbol{x})\partial_{x_j}^j$ with $\deg(\tilde{q}_j) \leqslant s$.

▷ *Find* $L' \in \mathfrak{Ann}(f\mathbb{1}_G) \cap \mathbb{K}[\boldsymbol{x}]\langle\partial_{x_i}\rangle$

1: $h_j \leftarrow \sum_{|\boldsymbol{\gamma}|\leqslant s} h_{j\boldsymbol{\gamma}}\boldsymbol{x}^{\boldsymbol{\gamma}}$ for $j \in [\![0, r]\!]$ with symbolic coefficients $h_{j\boldsymbol{\gamma}}$

2: $L' \leftarrow \sum_{j=0}^{r} h_j(\boldsymbol{x})\partial_{x_i}^j$

3: Find a nontrivial solution $\{h_{j\boldsymbol{\gamma}}\}$ of the linear system:

$$(L'^{\mathscr{M}} \cdot m)_{\boldsymbol{\alpha}} = 0, \qquad |\boldsymbol{\alpha}| \leqslant N$$

▷ *Extract minimal* $L \in \mathfrak{Ann}(f) \cap \mathbb{K}[\boldsymbol{x}]\langle\partial_{x_i}\rangle$

4: $\ell \leftarrow \mathrm{GCD}(h_0, \ldots, h_r)$ and $\tilde{q}_j \leftarrow h_j/\ell$ for $j \in [\![1, n]\!]$.

5: **return** $\tilde{L} = \sum_{j=0}^{r} \tilde{q}_j(\boldsymbol{x})\partial_{x_j}^j$

The coefficients $\{\tilde{q}_0, \ldots, \tilde{q}_r\}$ of the returned operator $\tilde{L} = \sum \tilde{q}_j\partial_{x_i}^j$ form a coprime family (line 4). This is also true for $\{q_0, \ldots, q_r\}$ by minimality of $\deg(q_r)$. By minimality of r, we have $\tilde{q}_r L - q_r\tilde{L} = 0$, that is $\tilde{q}_r q_j = q_r\tilde{q}_j$ for all j. Since $\mathbb{K}[\boldsymbol{x}]$ has the unique factorization property, there exists $\lambda \in \mathbb{K}$ s.t. $\tilde{q}_r = \lambda q_r$, yielding $\tilde{L} = \lambda L$. □

Reconstructing the support

From now on, we assume that a rectangular system $\{L_1, \ldots, L_n\}$ for the density f is known, and that L_i have the same order r.[1] Let:

$$L_i = \sum_{j=0}^{r} q_{i,j}(\boldsymbol{x})\partial_{x_i}^j \in \mathfrak{Ann}(f) \cap \mathbb{K}[\boldsymbol{x}]\langle\partial_{x_i}\rangle, \qquad i \in [\![1, n]\!].$$

The next assumption is crucial for support reconstruction. Roughly speaking, the differential system must not be singular over the Zariski closure of ∂G, except for a zero-measure set.

Assumption 9.22 *The pair* $\{g, q_{i,r}\}$ *is coprime for each* $i \in [\![1, n]\!]$.

Theorem 9.23 proves that **Algorithm** ReconstructSupport is correct.

Theorem 9.23 *Let analytic* f *be annihilated by the order* r *rectangular system* $\{L_1, \ldots, L_n\}$, *and* G *be as in* ***Assumption 9.9*** *with* $g \in \mathbb{K}[\boldsymbol{x}]$ *of degree* d. *Assume also* ***Assumption 9.22***. *Then, for* $N \geqslant \hat{N} := (2r-1)d + (d-1)b + s$, *with* $b = r \mod 2$ *and* $s = \max\{q_{i,r}\}$, ReconstructSupport$(n, d, r, \{L_i\}, N, (m_{\boldsymbol{\alpha}}))$ *returns* $\tilde{g} = \lambda g$ *with* $\lambda \in \mathbb{K}^*$. *In particular, this proves that when the density is known, the support can be reconstructed using moments up to degree* $(3r-1)d + (d-1)b + s + \max_{ij}\{\deg(q_{i,j}) - j\}$.

[1] Indeed, if L_i has order $r_i < r$, then it is replaced by $\partial_{x_i}^{r-r_i}L_i$, which has order r and the same leading polynomial coefficient.

Algorithm 9.4 ReconstructSupport$(n, d, r, \{L_i\}_{i=1}^n, N, (m_{\boldsymbol{\alpha}}))$

Input: $n \geqslant 2$, degree d, order r, rectangular system $\{L_1, \dots, L_n\}$ of order r, moments $m_{\boldsymbol{\alpha}}$ for $|\boldsymbol{\alpha}| \leqslant N + dr + \max_{ij}\{\deg(q_{i,j}) - j\}$.

Output: polynomial $\tilde{g}(\boldsymbol{x}) \in \mathbb{K}[\boldsymbol{x}]_d$ vanishing over ∂G.

1: $h \leftarrow \sum_{|\boldsymbol{\gamma}| \leqslant dr} h_{\boldsymbol{\gamma}} \boldsymbol{x}^{\boldsymbol{\gamma}}$ with symbolic coefficients $h_{\boldsymbol{\gamma}}$

2: Find a nontrivial solution $\{h_{\boldsymbol{\gamma}}\}$ of the linear system:

$$\left((hL_i)^{\mathscr{M}} \cdot m\right)_{\boldsymbol{\alpha}} = 0, \qquad |\boldsymbol{\alpha}| \leqslant N, i \in [\![1, n]\!]$$

3: **return** $\tilde{g} = h / \operatorname{GCD}(h, h_{x_1}, \dots, h_{x_n})$

Proof. First $h = g^r$ satisfies the linear system in line 2 since $g^r L_i \in \mathfrak{Ann}(f\mathbb{1}_G)$ by **Proposition 9.16**. Let h be any nontrivial solution, then $\langle hL_i \cdot (f\mathbb{1}_G), \varphi\rangle = 0$ for all $i \in [\![1, n]\!]$ and $\varphi \in \mathbb{K}[\boldsymbol{x}]_N$. Using **Proposition 9.15** combined with $L_i \cdot f = 0$, we get

$$\int_{\partial G} \mathcal{L}_{L_i}(f, h\varphi) \cdot \boldsymbol{n} \, \mathrm{d}S = 0, \qquad i \in [\![1, n]\!], \ \varphi \in \mathbb{K}[\boldsymbol{x}]_N.$$

Since L_i involves derivatives only in x_i, we have $\mathcal{L}_{L_i}(f, h\varphi) = \mathcal{L}_{L_i,i}(f, h\varphi)\boldsymbol{e_i}$, with the Lagrange bilinear concomitant [122]:

$$\begin{aligned} \mathcal{L}_{L_i,i}(f, h\varphi) = f &\left[q_{i,1}h\varphi - \partial_{x_i}(q_{i,2}h\varphi) + \cdots + (-1)^{r-1}\partial_{x_i}^{r-1}(q_{i,r}h\varphi)\right] \\ &+\partial_{x_i}(f)\left[q_{i,2}h\varphi - \partial_{x_i}(q_{i,3}h\varphi) + \cdots + (-1)^{r-2}\partial_{x_i}^{r-2}(q_{i,r}h\varphi)\right] \\ &+\dots \\ &+\partial_{x_i}^{r-1}(f)\, q_{i,r}h\varphi. \end{aligned} \tag{9.13}$$

We prove $h = \lambda g^r$ for some $\lambda \in \mathbb{K}^*$ by induction for k from 0 to r, showing $h = g^k h_k$ with $h_k \in \mathbb{R}[\boldsymbol{x}]_{(r-k)d}$. Of course this is true for $k = 0$ with $h_0 = h$. Now suppose that $h = g^k h_k$ for some $k < r$. Then let

$$\varphi = q_{i,r} h_k g^{r-1-k} g_{x_i}^b \in \mathbb{K}[\boldsymbol{x}]_{(2r-2k-1)d+(d-1)b+s} \subseteq \mathbb{K}[\boldsymbol{x}]_N,$$

Since $h\varphi$ is a multiple of g^{r-1}, all the terms in (9.13) are multiples of g (hence they vanish over ∂G), except for the derivative of order $r-1$, which we can write as

$$\partial_{x_i}^{r-1}(q_{i,r}h\varphi) = (r-1)! g_{x_i}^{r-1+b} q_{i,r}^2 h_k^2 + \ell(\boldsymbol{x}) g(\boldsymbol{x}), \qquad \ell(\boldsymbol{x}) \in \mathbb{K}[\boldsymbol{x}].$$

Therefore, integrating $\mathcal{L}_{L_i,i}(f, h\varphi)\ \boldsymbol{e_i} \cdot \boldsymbol{n} \, \mathrm{d}S$ over ∂G gives

$$\int_{\partial G} \left(g_{x_i}^{\frac{r+b}{2}} q_{i,r} h_k\right)^2 \frac{f}{\|\nabla g\|} \mathrm{d}S = 0,$$

implying that the squared polynomial in the integrand vanishes over ∂G, hence is a multiple of g. But g and $q_{i,r}$ are coprime by **Assumption 9.22**, so that g divides $h_k g_{x_i}$, for all $i \in [\![1, n]\!]$. Finally, since $\{g, g_{x_1}, \dots, g_{x_n}\}$ is a coprime family, g divides h_k, so $h_k = g h_{k+1}$. Now that $h = \lambda g^r$, $\operatorname{GCD}(h, h_{x_1}, \dots, h_{x_n}) = g^{r-1}$ (again since $\{g, g_{x_1}, \dots, g_{x_n}\}$ is coprime), so $\tilde{g} = \lambda g$. □

9.5 Examples and Conclusion

Our methods are exemplified in the two dimensional case, with respect to Lebesgue and restricted Gaussian measures[2]. The implementation uses `OreAlgebra` and `OreGroebnerBasis` routines from the HolonomicFunctions library [140]. The exactly computed moments m_{ij} (obtained from the recurrences given by **Algorithm** RecurrencesMoments together with closed-form initial conditions, when possible) are truncated to $\tilde{m}_{ij}$, s.t. $\lfloor -\log_{10} \frac{m_{i,j}-\tilde{m}_{ij}}{m_{ij}} \rfloor = \varepsilon$ i.e., ε represents the number of correct digits of $\tilde{m}_{ij}$. Then, given $\tilde{m}_{ij}$, **Algorithm** ReconstructExpPoly solves the inverse problem. The resulting overdetermined linear systems are solved numerically by a Least Mean Squares method of Mathematica.

Example 9.24 (Algebraic support, Lebesgue measure) *Consider the moments* $m_{ij} = \int_G x^i y^j \, dx\, dy$, *with respect to the Lebesgue measure, with* G *depicted with the checkered pattern in* ***Figure 9.1****.*

(i) Direct problem*: Given* $g = (x^2+y^2-1)(x^2+y^2-9)(x^2+(y-2)^2-1)((x-2)^2+y^2-1)$, *which vanishes on* ∂G, *and* $\mathfrak{Ann}\{1\} = \{\partial_x, \partial_y\}$, ***Algorithm*** RecurrencesMoments *returns a Gröbner basis with 9 generators and with* 36 monomials under the staircase*:* $\{S_i^k S_j^l,\ k,l \in \mathbb{N}, k+l \leqslant 7\}$.

(ii) Inverse problem*: Given a finite number of numerically computed moments* $\tilde{m}_{ij}$ *of the Lebesgue measure with unknown support* G*, the goal is to reconstruct* $g = \sum_{i+j\leqslant 8} g_{ij} x^i y^j$ *which vanishes on* ∂G*. The results of* ***Algorithm*** ReconstructExpPoly$(2, 8, 0, 22, (\tilde{m}_{ij})_{|i+j|\leqslant 29})$ *are depicted in* ***Figure 9.1****: the reconstructed boundary cannot be distinguished from the exact at the drawing scale, when the moments* $\tilde{m}_{ij}$ *are given with more than* 4 *correct digits. When* $2 \leqslant \varepsilon \leqslant 4$*, the actual geometric boundary of* G*, can still be very well reconstructed, although the algebraic boundary is degraded.*

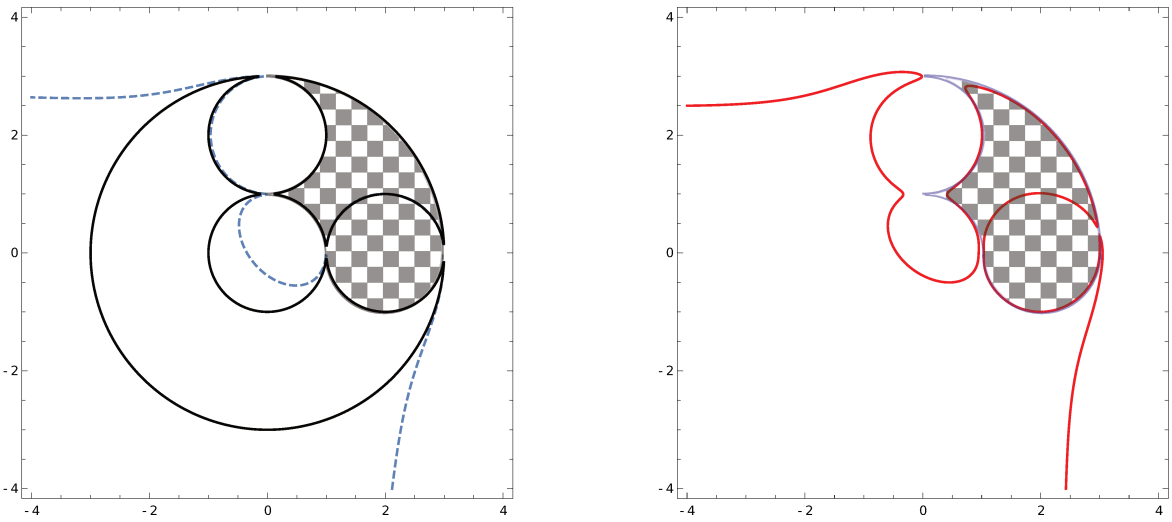
Example 9.25 (Algebraic Support, Gaussian measure) *Consider the moments* $m_{ij} = \int_G x^i y^j \exp(p(x,y)) dx\, dy$*. In* ***Figure 9.2****(a),* G *is checkered and the level curves of* $\exp(p(x,y))$ *are in dashed.*

(i) Direct problem*: Given* $g = \left(x^2 - 9/10\right)^2 + \left(x^2 - 11/10\right)^2 - 1$, *which vanishes on* ∂G, *and* $f = \exp\left(-x^2 + xy - y^2/2\right)$, *with* $\mathfrak{Ann}\{f\} = \{\partial_x + 2x - 1, \partial_y + y - 1\}$, *apply* ***Algorithm*** RecurrencesMoments *to compute a Gröbner basis for the sequence of moments* m_{ij}*. In the same setting as above, a Gröbner basis with 5 generators and with* 28 monomials under the staircase *is obtained.*

(ii) Inverse problem*: Suppose now given a finite number of numerically computed moments* $\tilde{m}_{ij}$*, with unknown support* G *and unknown Gaussian weight. The goal is to reconstruct* $g = \sum_{i+j\leqslant 4} g_{ij} x^i y^j$ *which vanishes on* ∂G*, as well as* $p = \sum_{i+j\leqslant 2} p_{ij} x^i y^j$*.* ***Algorithm*** ReconstructExpPoly *called with parameters* $(2, 4, 2, 14, (\tilde{m}_{ij})_{|i+j|\leqslant 18})$ *provides the reconstructed* g*, as depicted in* ***Figure 9.2****(b): the reconstructed boundary cannot be distinguished from the exact at the drawing scale, when* $\varepsilon > 8$*. When* $4 \leqslant \varepsilon \leqslant 8$*, the actual geometric boundary of* G*, can still be very well reconstructed. Concerning the Gaussian weight, the situation is similar, cf.* ***Figure 9.2****(c).*

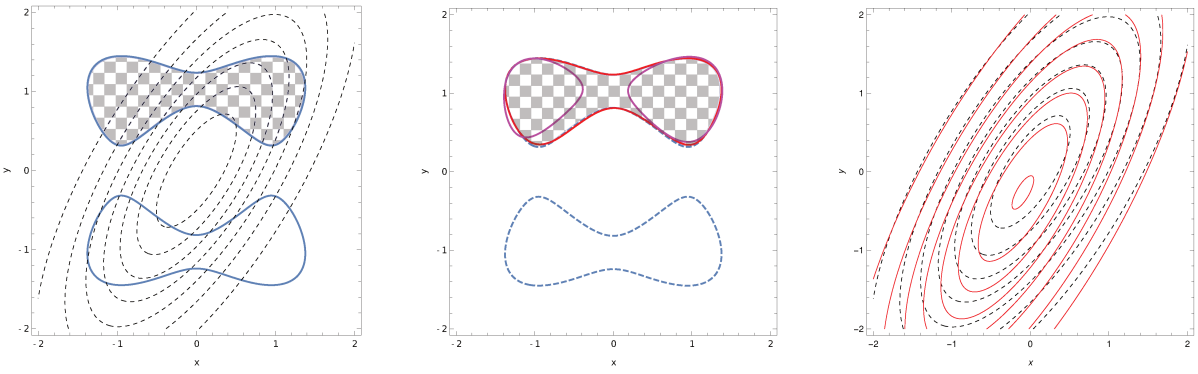
[2] The code is available at `http://homepages.laas.fr/fbrehard/HolonomicMomentProblem`

The proposed method is very robust on the above academic examples, but a further study is needed for an efficient implementation in practical higher-dimensional cases. On the theoretical side, this chapter provides further insight on the question raised in **[150]** regarding the *finite determinateness of a measure*. To sum up, provided **Assumptions 9.9** and **9.22** hold, for a measure with compact algebraic support G, with $g \in \mathbb{R}[\boldsymbol{x}]_d$ vanishing on ∂G and known holonomic density f, the moments up to degree N (which only depends on d and the order of a rectangular differential system which annihilates f) determine in a constructive and robust manner the coefficients of g. Thus, this determines in turn all the other moments. When both the density and the support are unknown, a uniform bound N does not exist in general. We provided the solution for the special case of unknown *exponential-polynomial* density.



(a) (b)

Figure 9.1: (a) G in checkered pattern, together with ∂G in black. For $\varepsilon > 4$: reconstructed and original boundary cannot be distinguished at this scale; in dashed-blue, $\varepsilon = 4$, while in red (b) $\varepsilon = 2$.



(a) (b) (c)

Figure 9.2: (a) G in checkered pattern, exact Gaussian level curves in dashed-black, ∂G in blue; (b) When $\varepsilon > 8$: reconstructed and original boundary (in dashed blue) cannot be distinguished at this scale; in red, $\varepsilon = 6$, and in magenta for $\varepsilon = 4$ digits. (c) Reconstructed Gaussian level curves in red when $\varepsilon = 8$; when $\varepsilon > 8$, the level curves of exact and reconstructed coincide at this scale.

Bibliography

[1] Sergei A. Abramov, Moulay A. Barkatou, and Mark Van Hoeij. Apparent singularities of linear difference equations with polynomial coefficients. *Appl. Algebra Eng. Commun. Comput.*, 17(2):117–133, 2006.

[2] Milton Abramowitz and Irene A. Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, volume 55 of *National Bureau of Standards Applied Mathematics Series*. Courier Corporation, 1964.

[3] Ravi P. Agarwal. Contraction and approximate contraction with an application to multipoint boundary value problems. *J. Comput. Appl. Math.*, 9(4):315–325, 1983.

[4] Edward Anderson, Zhaojun Bai, Christian Bischof, Susan Blackford, Jack Dongarra, Jeremy Du Croz, Anne Greenbaum, Sven Hammarling, Alan McKenney, and Danny Sorensen. *LAPACK Users' guide*, volume 9. Siam, 1999.

[5] Paulo Ricardo Arantes Gilz. *Embedded and validated control algorithms for the spacecraft rendezvous*. PhD thesis, Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier), October 2018.

[6] Paulo Ricardo Arantes Gilz, Florent Bréhard, and Clément Gazzino. Validated Semi-Analytical Transition Matrix for Linearized Relative Spacecraft Dynamics via Chebyshev Polynomials. In *2018 Space Flight Mechanics Meeting, AIAA Science and Technology Forum and Exposition*, page 24, 2018.

[7] Paulo Ricardo Arantes Gilz, Mioara Joldes, Christophe Louembet, and Frédéric Camps. Model predictive control for rendezvous hovering phases based on a novel description of constrained trajectories. In *Proceedings of the 20th IFAC World Congress, Toulouse*, pages 7490–7495, July 2017.

[8] Paulo Ricardo Arantes Gilz and Christophe Louembet. Predictive control algorithm for spacecraft rendezvous hovering phases. In *Control Conference (ECC), 2015 European*, pages 2085–2090. IEEE, 2015.

[9] Gianni Arioli and Hans Koch. Integration of dissipative partial differential equations: a case study. *SIAM Journal on Applied Dynamical Systems*, 9(3):1119–1133, 2010.

[10] V. I. Arnol'd. Loss of stability of self-induced oscillations near resonance, and versal deformations of equivariant vector fields. *Funkcional. Anal. i Priložen.*, 11(2):1–10, 95, 1977. Functional Anal. Appl. **11** (1977), no. 2, 85–92.

[11] V. I. Arnol'd. *Ten problems*, volume 1 of *Adv. Soviet Math.* Amer. Math. Soc., Providence, RI, 1990.

[12] Denis Arzelier, Florent Bréhard, Norbert Deak, Mioara Joldes, Christophe Louembet, Aude Rondepierre, and Romain Serra. Linearized impulsive fixed-time fuel-optimal space rendezvous: A new numerical approach. *IFAC-PapersOnLine*, 49(17):373–378, 2016.

[13] Denis Arzelier, Florent Bréhard, and Mioara Joldes. Exchange algorithm for evaluation and approximation error-optimized polynomials. In *26th IEEE Symposium on Computer Arithmetic (ARITH-26)*. IEEE, 2019. To appear soon.

[14] Roberto Barrio, Hao Jiang, and Sergio Serrano. A general condition number for polynomials. *SIAM J. Numer. Anal.*, 51(2):1280–1294, 2013.

[15] Dmitry Batenkov. Moment inversion problem for piecewise D-finite functions. *Inverse Problems*, 25(10):105001, 24, 2009.

[16] Richard H. Battin. *An introduction to the mathematics and methods of astrodynamics*. AIAA Education Series. American Institute of Aeronautics and Astronautics (AIAA), Reston, VA, revised edition, 1999. With a foreword by J. S. Przemieniecki.

[17] Bernhard Beckermann. The condition number of real Vandermonde, Krylov and positive definite Hankel matrices. *Numer. Math.*, 85(4):553–577, 2000.

[18] Alexandre Benoit. *Algorithmique semi-numérique rapide des séries de Tchebychev*. PhD thesis, École Polytechnique, 2012.

[19] Alexandre Benoit, Mioara Joldeș, and Marc Mezzarobba. Rigorous uniform approximation of D-finite functions using Chebyshev expansions. *Math. Comp.*, 86(305):1303–1341, 2017.

[20] Vasile Berinde. *Iterative approximation of fixed points*, volume 1912 of *Lecture Notes in Mathematics*. Springer, Berlin, 2007.

[21] I. N. Bernšteĭn. Modules over a ring of differential operators. An investigation of the fundamental solutions of equations with constant coefficients. *Funkcional. Anal. i Priložen.*, 5(2):1–16, 1971.

[22] Yves Bertot and Pierre Castéran. *Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions*. Springer Science & Business Media, 2013.

[23] M. Berz and K. Makino. Suppression of the wrapping effect by Taylor model-based verified integrators: Long-term stabilization by shrink wrapping. *Int. J. Diff. Eq. Appl.*, 10:385–403, 2005.

[24] Martin Berz and Kyoko Makino. Rigorous global search using Taylor models. In *Proceedings of the 2009 conference on Symbolic numeric computation*, pages 11–20. ACM, 2009.

[25] John T. Betts. Survey of numerical methods for trajectory optimization. *J. Guidance Control Dynam.*, 21(2):193–207, 1998.

[26] Gal Binyamini, Dmitry Novikov, and Sergei Yakovenko. On the number of zeros of Abelian integrals. *Invent. Math.*, 181(2):227–289, 2010.

[27] Sylvie Boldo, François Clément, Florian Faissole, Vincent Martin, and Micaela Mayero. A Coq formal proof of the Lax–Milgram theorem. In *6th ACM SIGPLAN Conference on Certified Programs and Proofs*, Paris, France, January 2017.

[28] Sylvie Boldo, Florian Faissole, and Alexandre Chapoutot. Round-off error analysis of explicit one-step numerical integration methods. In *2017 IEEE 24th Symposium on Computer Arithmetic (ARITH)*, pages 82–89. IEEE, 2017.

[29] Sylvie Boldo, Catherine Lelay, and Guillaume Melquiond. Coquelicot: a user-friendly library of real analysis for Coq. *Math. Comput. Sci.*, 9(1):41–62, 2015.

[30] Sylvie Boldo and Guillaume Melquiond. Flocq: A unified library for proving floating-point algorithms in Coq. In *2011 IEEE 20th Symposium on Computer Arithmetic*, pages 243–252. IEEE, 2011.

[31] Arindam Bose. Did You Know : The History of Egyptian Mathematics (Part II) – Egyptian Numerals. `http://arindambose.com/?p=737`, 2015.

[32] Alin Bostan, Frédéric Chyzak, Pierre Lairez, and Bruno Salvy. Generalized Hermite reduction, creative telescoping and definite integration of D-finite functions. In *Proceedings of International Symposium on Symbolic and Algebraic Computation, New York, USA,2018*, pages 95–102, 2018.

[33] Nicolas Bourbaki. *General Topology*. Springer, 1995. Original French edition published by Masson, Paris, 1971.

[34] John P. Boyd. *Chebyshev and Fourier spectral methods*. Dover Publications, 2001.

[35] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[36] Thierry Braconnier and Philippe Langlois. From rounding error estimation to automatic correction with automatic differentiation. In *Automatic differentiation of algorithms*, pages 351–357. Springer, 2002.

[37] Florent Bréhard. A Newton-like validation method for chebyshev approximate solutions of linear ordinary differential systems. In *ISSAC 2018-43rd International Symposium on Symbolic and Algebraic Computation*, pages 103–110. ACM, 2018.

[38] Florent Bréhard, Nicolas Brisebarre, and Mioara Joldes. Validated and numerically efficient chebyshev spectral methods for linear ordinary differential equations. *ACM Trans. Math. Software*, 44(4):44:1–44:42, July 2018.

[39] Florent Bréhard, Mioara Joldes, and Jean-Bernard Lasserre. On a moment problem with holonomic functions. In *44th International Symposium on Symbolic and Algebraic Computation (ISSAC 2019)*. ACM, 2019. To appear soon.

[40] Florent Bréhard, Assia Mahboubi, and Damien Pous. A certificate-based approach to formally verified approximations, 2019. Submitted.

[41] Haim Brezis. *Functional analysis, Sobolev spaces and partial differential equations*. Springer Science & Business Media, 2010.

[42] N. Brisebarre, J.-M. Muller, and A. Tisserand. Computing machine-efficient polynomial approximations. *ACM Trans. Math. Software*, 32(2):236–256, June 2006.

[43] Nicolas Brisebarre and Sylvain Chevillard. Efficient polynomial L^∞ approximations. In *18th IEEE Symposium on Computer Arithmetic (ARITH-18)*, pages 169–176, Montpellier, France, 2007.

[44] Nicolas Brisebarre and Mioara Joldeș. Chebyshev interpolation polynomial-based tools for rigorous computing. In *Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation*, pages 147–154. ACM, 2010.

[45] Nicolas Brisebarre, Mioara Joldeș, Jean-Michel Muller, Ana-Maria Naneș, and Joris Picot. Error analysis of some operations involved in the fast Fourier transform. `https://hal.archives-ouvertes.fr/hal-01949458/`, 2018.

[46] G. Campan and P. Brousse. ORANGE: Orbital analytical model for geosynchronous satellite. *Revista Brasileira de Ciencias Mecanicas (ISSN 0100-7386), vol. 16, p. 561-572*, 16:561–572, 1994.

[47] E. Capello, F. Dabbene, G. Guglieri, and E. Punta. "Flyable" Guidance and Control Algorithms for Orbital Rendezvous Maneuver. *SICE Journal of Control, Measurement, and System Integration*, 11(1):14–24, 2018.

[48] C. Carasso and P. J. Laurent. Un algorithme général pour l'approximation au sens de Tchebycheff de fonctions bornées sur un ensemble quelconque. In *Approximation Theory*, number 556 in Lecture Notes in Mathematics, pages 99–121. Springer Berlin Heidelberg, 1976.

[49] Claude Carasso. *L'algorithme d'échange en optimisation convexe*. PhD thesis, Université Joseph-Fourier-Grenoble I, 1973.

[50] Bruce L. Chalmers. The Remez exchange algorithm for approximation with linear restrictions. *Trans. Amer. Math. Soc.*, 223:103–131, 1976.

[51] Fengde Chen, Chengzhi Li, Jaume Llibre, and Zenghua Zhang. A unified proof on the weak Hilbert 16th problem for $n = 2$. *J. Differential Equations*, 221(2):309–342, 2006.

[52] Lan Sun Chen and Ming Shu Wang. The relative position, and the number, of limit cycles of a quadratic differential system. *Acta Math. Sinica*, 22(6):751–758, 1979.

[53] Elliott Ward Cheney. *Introduction to approximation theory*. AMS Chelsea Publishing, Providence, RI, 1998. Reprint of the second (1982) edition.

[54] Sylvain Chevillard, John Harrison, Mioara Joldeș, and Christoph Lauter. Efficient and accurate computation of upper bounds of approximation errors. *Theoret. Comput. Sci.*, 412(16):1523–1543, 2011.

[55] Sylvain Chevillard, Mioara Joldeș, and Christoph Lauter. Sollya: An environment for the development of numerical codes. In *International Congress on Mathematical Software*, pages 28–31. Springer, 2010.

[56] Colin Christopher. Estimating limit cycle bifurcations from centers. In *Differential equations with symbolic computation*, Trends Math., pages 23–35. Birkhäuser, Basel, 2005.

[57] Colin Christopher and Chengzhi Li. *Limit cycles of differential equations*. Advanced Courses in Mathematics. CRM Barcelona. Birkhäuser Verlag, Basel, 2007.

[58] Frédéric Chyzak. *Fonctions holonomes en calcul formel*. PhD thesis, Ecole Polytechnique X, 1998.

[59] Frédéric Chyzak. *The ABC of Creative Telescoping: Algorithms, Bounds, Complexity*. Memoir of accreditation to supervise research (HDR), Université d'Orsay, April 2014.

[60] C. W. Clenshaw. A note on the summation of Chebyshev series. *Math. Tables Aids Comput.*, 9:118–120, 1955.

[61] C. W. Clenshaw. The numerical solution of linear differential equations in Chebyshev series. *Proc. Cambridge Philos. Soc.*, 53:134–149, 1957.

[62] Earl A. Coddington and Norman Levinson. *Theory of ordinary differential equations*. McGraw-Hill Book Company, Inc., New York-Toronto-London, 1955.

[63] Luís Cruz-Filipe, Herman Geuvers, and Freek Wiedijk. C-corn, the constructive coq repository at nijmegen. In *International Conference on Mathematical Knowledge Management*, pages 88–103. Springer, 2004.

[64] C. Daramy-Loirat, D. Defour, F. de Dinechin, M. Gallet, N. Gast, C. Q. Lauter, and J.-M. Muller. CR-LIBM, A library of correctly-rounded elementary functions in double-precision. Technical report, LIP Laboratory, Arenaire team, December 2006.

[65] Eva Darulova and Viktor Kuncak. Towards a compiler for reals. *ACM Trans. Program. Lang. Syst.*, 39(2):8:1–8:28, March 2017.

[66] Marc Daumas and Guillaume Melquiond. Certification of bounds on expressions involving rounded operators. *ACM Trans. Math. Software*, 37(1):Art. 2, 20, 2010.

[67] Florent de Dinechin, Christoph Lauter, and Guillaume Melquiond. Certifying the floating-point implementation of an elementary function using Gappa. *IEEE Trans. Comput.*, 60(2):242–253, 2011.

[68] Florent de Dinechin, Christoph Q. Lauter, and J.-M. Muller. Fast and correctly rounded logarithms in double-precision. *Theor. Inform. Appl.*, 41(1):85–102, 2007.

[69] G. Deaconu. *On the trajectory design, guidance and control for spacecraft rendezvous and proximity operations*. PhD thesis, Univ. Toulouse 3 - Paul Sabatier, Toulouse, France, October 2013.

[70] Georgia Deaconu, Christophe Louembet, and Alain Théron. Constrained periodic spacecraft relative motion using non-negative polynomials. In *American Control Conference (ACC), 2012*, pages 6715–6720. IEEE, 2012.

[71] Giuseppe Di Mauro, Markus Schlotterer, Stephan Theil, and Michèle Lavagna. Nonlinear control for proximity operations based on differential algebra. *J. Guidance Control Dynam.*, 38(11):2173–2187, 2015.

[72] T. A Driscoll, N. Hale, and L. N. Trefethen. *Chebfun Guide*. Pafnuty Publications, 2014.

[73] Kui Du. On well-conditioned spectral collocation and spectral methods by the integral reformulation. *SIAM J. Sci. Comput.*, 38(5):A3247–A3263, 2016.

[74] H. Dulac. Sur les cycles limites. *Bull. Soc. Math. France*, 51:45–188, 1923.

[75] T. Dzetkulič. Rigorous integration of non-linear ordinary differential equations in Chebyshev basis. *Numer. Algorithms*, 69:183–205, 2015.

[76] Peter Ebenfelt, Björn Gustafsson, Dmitry Khavinson, and Mihai Putinar, editors. *Quadrature domains and their applications*, volume 156 of *Operator Theory: Advances and Applications*. Birkhäuser Verlag, Basel, 2005.

[77] Jean Écalle. *Introduction aux fonctions analysables et preuve constructive de la conjecture de Dulac*. Actualités Mathématiques. Hermann, Paris, 1992.

[78] 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.

[79] C. Epstein, W. L. Miranker, and T. J. Rivlin. Ultra-arithmetic. I. Function data types. *Math. Comput. Simulation*, 24(1):1–18, 1982.

[80] C. Epstein, W. L. Miranker, and T. J. Rivlin. Ultra-arithmetic. II. Intervals of polynomials. *Math. Comput. Simulation*, 24(1):19–29, 1982.

[81] Wigbert Fehse. *Automated rendezvous and docking of spacecraft*, volume 16. Cambridge University Press, 2003.

[82] Silviu-Ioan Filip. *Robust tools for weighted Chebyshev approximation and applications to digital filter design*. PhD thesis, École Normale Supérieure de Lyon, 2016.

[83] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann. MPFR: A Multiple-Precision Binary Floating-Point Library with Correct Rounding. *ACM Trans. Math. Software*, 33(2), 2007. Available at `http://www.mpfr.org/`.

[84] L. Fox and I. B. Parker. *Chebyshev polynomials in numerical analysis*. Oxford University Press, London-New York-Toronto, Ont., 1968.

[85] André Galligo. Some algorithmic questions on ideals of differential operators. In *European Conference on Computer Algebra*, pages 413–421. Springer, 1985.

[86] C. Gazzino. Dynamics of a Geostationary Satellite. Technical report, LAAS-CNRS, hal-01644934, 2017.

[87] C. Gazzino, D. Arzelier, L. Cerri, D. Losa, C. Louembet, and C. Pittet. Solving the Minimum-Fuel Low-Thrust Geostationary Station Keeping Problem via the Switching Systems Theory. In *European Conference for Aeronautics and AeroSpace Sciences, EUCASS2017*, Milano, Italy, 2017.

[88] C. Gazzino, C. Louembet, D. Arzelier, N. Jozefowiez, D. Losa, C. Pittet, and L. Cerri. Integer Programming for Optimal Control of Geostationary Station Keeping of Low-Thrust Satellites. In *IFAC 2017 World Congress*, pages 8169–8174, Toulouse, France, 2017.

[89] Clément Gazzino. *Stratégies de maintien à poste pour un satellite géostationnaire à propulsion tout électrique*. PhD thesis, Université Paul Sabatier - Toulouse III, January 2018.

[90] Clément Gazzino, Denis Arzelier, Christophe Louembet, Luca Cerri, Christelle Pittet, and Damiana Losa. Long-term electric-propulsion geostationary station-keeping via integer programming. *J. Guidance Control Dynam.*, 42(5):976–991, 2019.

[91] Dong-Woo Gim and Kyle T Alfriend. Satellite relative motion using differential equinoctial elements. *Celestial Mechanics and Dynamical Astronomy*, 92(4):295–336, 2005.

[92] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 1989.

[93] Israel Gohberg, Seymour Goldberg, and Marinus A. Kaashoek. *Basic classes of linear operators*. Birkhäuser Verlag, Basel, 2003.

[94] Gene H. Golub, Peyman Milanfar, and James Varah. A stable numerical method for inverting shape from moments. *SIAM J. Sci. Comput.*, 21(4):1222–1243, 1999/00.

[95] Georges Gonthier. Formal proof—the four-color theorem. *Notices Amer. Math. Soc.*, 55(11):1382–1393, 2008.

[96] David Gottlieb and Steven A. Orszag. *Numerical Analysis of Spectral Methods: Theory and Applications*, volume 26. SIAM, 1977.

[97] Torbjörn Granlund. The GNU multiple precision arithmetic library, 1996. `https://gmplib.org`.

[98] Nick Gravin, Jean Lasserre, Dmitrii V. Pasechnik, and Sinai Robins. The inverse moment problem for convex polytopes. *Discrete & Comp. Geometry*, 48(3):596–621, 2012.

[99] Leslie Greengard. Spectral integration and two-point boundary value problems. *SIAM J. Numer. Anal.*, 28(4):1071–1080, 1991.

[100] Markus Grimmer, Knut Petras, and Nathalie Revol. Multiple precision interval packages: Comparing different approaches. In *Numerical Software with Result Verification*, pages 64–90. Springer, 2004.

[101] Thomas Hales, Mark Adams, Gertrud Bauer, Tat Dat Dang, John Harrison, Hoang Le Truong, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Tat Thang Nguyen, et al. A formal proof of the kepler conjecture. In *Forum of Mathematics, Pi*, volume 5. Cambridge University Press, 2017.

[102] Thomas C. Hales. A proof of the Kepler conjecture. *Ann. of Math. (2)*, 162(3):1065–1185, 2005.

[103] Royden Halsey and Fitzpatrick Patrick. *Real Analysis*. Prentice Hall, 2010.

[104] E. N. Hartley. A tutorial on model predictive control for spacecraft rendezvous. In *2015 European Control Conference (ECC)*, pages 1355–1361, July 2015.

[105] Erich Hecke. *Lectures on the theory of algebraic numbers*, volume 77 of *Graduate Texts in Mathematics*. Springer-Verlag, New York-Berlin, 1981. Translated from the German by George U. Brauer, Jay R. Goldman and R. Kotzen.

[106] Nicholas J. Higham. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

[107] 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.

[108] Johannes Hölzl, Fabian Immler, and Brian Huffman. Type classes and filters for mathematical analysis in Isabelle/HOL. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Interactive Theorem Proving*, pages 279–294, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[109] Karel Hrbacek and Thomas Jech. *Introduction to set theory, revised and expanded*. Crc Press, 1999.

[110] David G. Hull. Conversion of optimal control problems into parameter optimization problems. *J. Guidance Control Dynam.*, 20(1):57–60, 1997.

[111] Allan Hungria, Jean-Philippe Lessard, and Jason D. Mireles James. Rigorous numerics for analytic solutions of differential equations: the radii polynomial approach. *Math. Comp.*, 85(299):1427–1459, 2016.

[112] IEEE Computer Society. *IEEE Standard for Floating-Point Arithmetic*. IEEE Standard 754-1985, 1985. Available at `https://ieeexplore.ieee.org/document/30711`.

[113] IEEE Computer Society. *IEEE Standard for Floating-Point Arithmetic*. IEEE Standard 754-2008, August 2008. Available at `http://ieeexplore.ieee.org/servlet/opac?punumber=4610933`.

[114] Georges Ifrah. *The Universal History of Computing: From the Abacus to the Quantum Computer*. John Wiley & Sons, Inc., 2001.

[115] Georges Ifrah. *The Universal History of Numbers: From Prehistory to the Invention of the Computer*. John Wiley & Sons, Inc., 2001.

[116] Yu. Ilyashenko. Centennial history of Hilbert's 16th problem. *Bull. Amer. Math. Soc. (N.S.)*, 39(3):301–354, 2002.

[117] Yu. S. Il'yashenko. In the theory of normal forms of analytic differential equations violating the conditions of A. D. Bryuno divergence is the rule and convergence the exception. *Vestnik Moskov. Univ. Ser. I Mat. Mekh.*, (2):10–16, 86, 1981.

[118] Yu. S. Il'yashenko. *Finiteness theorems for limit cycles*, volume 94 of *Translations of Mathematical Monographs*. American Mathematical Society, Providence, RI, 1991. Translated from the Russian by H. H. McFaden.

[119] Fabian Immler. A verified ODE solver and the Lorenz attractor. *J. Automat. Reason.*, 61(1-4):73–111, 2018.

[120] Fabian Immler and Johannes Hölzl. Numerical analysis of ordinary differential equations in Isabelle/HOL. In *International Conference on Interactive Theorem Proving*, pages 377–392. Springer, 2012.

[121] G. Inalhan, M. Tillerson, and J. P. How. Relative dynamics and control of spacecraft formations in eccentric orbits. *J. Guidance Control Dynam.*, 25(1):48–59, January 2002.

[122] E. L. Ince. *Ordinary Differential Equations*. Dover Publications, New York, 1956.

[123] D. J. Irvin, R. G Cobb, and T. A. Lovell. Fuel-optimal maneuvers for constrained relative satellite orbits. *Journal of guidance, control, and dynamics*, 32(3):960–973, 2009.

[124] Arieh Iserles. *A first course in the numerical analysis of differential equations*. Cambridge Texts in Applied Mathematics. Cambridge University Press, Cambridge, second edition, 2009.

[125] Fredrik Johansson. Arb: efficient arbitrary-precision midpoint-radius interval arithmetic. *IEEE Trans. Comput.*, 66(8):1281–1292, 2017.

[126] Tomas Johnson. A quartic system with twenty-six limit cycles. *Exp. Math.*, 20(3):323–328, 2011.

[127] Tomas Johnson and Warwick Tucker. An improved lower bound on the number of limit cycles bifurcating from a Hamiltonian planar vector field of degree 7. *Internat. J. Bifur. Chaos Appl. Sci. Engrg.*, 20(5):1451–1458, 2010.

[128] Tomas Johnson and Warwick Tucker. An improved lower bound on the number of limit cycles bifurcating from a quintic Hamiltonian planar vector field under quintic perturbation. *Internat. J. Bifur. Chaos Appl. Sci. Engrg.*, 20(1):63–70, 2010.

[129] Mioara Joldeș. *Rigorous Polynomial Approximations and Applications*. PhD thesis, École normale supérieure de Lyon – Université de Lyon, Lyon, France, 2011.

[130] Mioara Joldes, Jean-Michel Muller, Valentina Popescu, and Warwick Tucker. Campary: cuda multiple precision arithmetic library and applications. In *International Congress on Mathematical Software*, pages 232–240. Springer, 2016.

[131] L. V. Kantorovich and G. P. Akilov. *Functional analysis*. Pergamon Press, Oxford-Elmsford, N.Y., second edition, 1982. Translated from the Russian by Howard L. Silcock.

[132] L. V. Kantorovich, B. Z. Vulikh, and A. G. Pinsker. Functional analysis in partially ordered spaces (in Russian). *Gostekhizdat, Moscow*, 1950.

[133] Samuel Karlin and William J. Studden. *Tchebycheff systems: With applications in analysis and statistics*. Pure and Applied Mathematics, Vol. XV. Interscience Publishers John Wiley & Sons, New York-London-Sydney, 1966.

[134] Yitzhak Katznelson. *An introduction to harmonic analysis*. Cambridge University Press, 2004.

[135] Edgar Kaucher. Solving function space problems with guaranteed close bounds. In *Proc. of the symposium on A new approach to scientific computation*, pages 139–164. Academic Press Professional, Inc., 1983.

[136] Edgar W. Kaucher and Willard L. Miranker. *Self-validating numerics for function space problems: Computation with guarantees for differential and integral equations*, volume 9. Elsevier, 1984.

[137] Gershon Kedem. A posteriori error bounds for two-point boundary value problems. *SIAM Journal on Numerical Analysis*, 18(3):431–448, 1981.

[138] A. G. Khovanskiĭ. Real analytic manifolds with the property of finiteness, and complex abelian integrals. *Funktsional. Anal. i Prilozhen.*, 18(2):40–50, 1984. Functional Anal. Appl. **18** (1984), no. 2, 119–127.

[139] Rudi Klatte, Ulrich Kulisch, Andreas Wiethoff, and Michael Rauch. *C-XSC: A C++ class library for extended scientific computing*. Springer Science & Business Media, 2012.

[140] Christoph Koutschan. Advanced applications of the holonomic systems approach. *ACM Comm. Computer Algebra*, 43(3/4):119, 2009.

[141] Christoph Koutschan. *Advanced applications of the holonomic systems approach*. PhD thesis, Research Institute for Symbolic Computation (RISC), Johannes Kepler University, Linz, Austria, 2009.

[142] Christoph Koutschan. Holonomic functions (user's guide), 2010. `https://www3.risc.jku.at/research/combinat/software/ergosum/RISC/HolonomicFunctions.html`.

[143] Ulrich Kulisch. An axiomatic approach to rounded computations. *Numer. Math.*, 18:1–17, 1971/72.

[144] Olga Kupriianova and Christoph Lauter. Metalibm: A mathematical functions code generator. In *International Congress on Mathematical Software*, pages 713–717. Springer, 2014.

[145] Traian Lalescu. *Introduction à la théorie des équations intégrales (Introduction to the Theory of Integral Equations)*. Librairie Scientifique A. Hermann, 1911.

[146] E. M. Landis and I. G. Petrovskiĭ. A letter to the editors. *Mat. Sb. (N.S.)*, 73 (115):160, 1967.

[147] Oscar E. Lanford, III. A computer-assisted proof of the Feigenbaum conjectures. *Bull. Amer. Math. Soc. (N.S.)*, 6(3):427–434, 1982.

[148] Jean Bernard Lasserre. *Moments, positive polynomials and their applications*, volume 1 of *Imperial College Press Optimization Series*. Imperial College Press, London, 2010.

[149] Jean Bernard Lasserre. Recovering an homogeneous polynomial from moments of its level set. *Discrete Comput. Geom.*, 50(3):673–678, 2013.

[150] Jean Bernard Lasserre and Mihai Putinar. Algebraic-exponential data recovery from moments. *Discrete Comput. Geom.*, 54(4):993–1012, 2015.

[151] C. Q. Lauter. *Arrondi Correct de Fonctions Mathématiques.* PhD thesis, ÉNS de Lyon, Lyon, France, October 2008.

[152] Christoph Lauter and Marc Mezzarobba. Semi-automatic floating-point implementation of special functions. In *2015 IEEE 22nd Symposium on Computer Arithmetic*, pages 58–65. IEEE, 2015.

[153] Jean-Philippe Lessard and Christian Reinhardt. Rigorous numerics for nonlinear differential equations using Chebyshev series. *SIAM J. Numer. Anal.*, 52(1):1–22, 2014.

[154] Chengzhi Li, Changjian Liu, and Jiazhong Yang. A cubic system with thirteen limit cycles. *J. Differential Equations*, 246(9):3609–3619, 2009.

[155] Ji Bin Li and Qi Ming Huang. Bifurcations of limit cycles forming compound eyes in the cubic system. *Chinese Ann. Math. Ser. B*, 8(4):391–403, 1987. A Chinese summary appears in Chinese Ann. Math. Ser. A 8 (1987), no. 5, 643.

[156] Jibin Li. Hilbert's 16th problem and bifurcations of planar polynomial vector fields. *Internat. J. Bifur. Chaos Appl. Sci. Engrg.*, 13(1):47–106, 2003.

[157] Guoqing Liu and Vladik Kreinovich. Fast convolution and fast Fourier transform under interval and fuzzy uncertainty. *J. Comput. System Sci.*, 76(1):63–76, 2010.

[158] P. Di Lizia, R. Armellin, and M. Lavagna. Application of high order expansions of two-point boundary value problems to astrodynamics. *Celestial Mech. Dynam. Astronom.*, 102(4):355–375, 2008.

[159] P. Di Lizia, R. Armellin, A. Morselli, and F. Bernelli-Zazzera. High order optimal feedback control of space trajectories with bounded control. *Acta Astronautica*, 94(1):383 – 394, 2014.

[160] Johan Löfberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*, pages 284–289. IEEE, 2004.

[161] D. Losa. *High vs low thrust station keeping maneuver planning for geostationary satellites.* PhD thesis, École Nationale des Mines de Paris, 2007.

[162] Nicolas Magaud and Yves Bertot. Changing data structures in type theory: A study of natural numbers. In *International Workshop on Types for Proofs and Programs*, pages 181–196. Springer, 2000.

[163] K. Makino and M. Berz. Suppression of the wrapping effect by Taylor model-based verified integrators: Long-term stabilization by preconditioning. *Int. J. Diff. Eq. Appl.*, 10:353–384, 2005.

[164] Kyoko Makino. *Rigorous analysis of nonlinear motion in particle accelerators.* PhD thesis, Michigan State University. Department of Physics and Astronomy, 1998.

[165] Kyoko Makino and Martin Berz. Taylor models and other validated functional inclusion methods. *Int. J. Pure Appl. Math.*, 4(4):379–456, 2003.

[166] Kyoko Makino and Martin Berz. Cosy infinity version 9. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 558(1):346–350, 2006.

[167] Kyoko Makino and Martin Berz. Suppression of the wrapping effect by Taylor model-based verified integrators: the single step. *Int. J. Pure Appl. Math.*, 36(2):175–197, 2007.

[168] Érik Martin-Dorel and Guillaume Melquiond. Proving tight bounds on univariate expressions with elementary functions in coq. *Journal of Automated Reasoning*, 57(3):187–217, Oct 2016.

[169] Per Martin-Löf and Giovanni Sambin. *Intuitionistic type theory*, volume 9. Bibliopolis Naples, 1984.

[170] John C. Mason and David C. Handscomb. *Chebyshev polynomials*. CRC Press, 2002.

[171] Guillaume Melquiond. Proving bounds on real-valued functions with computations. In *International Joint Conference on Automated Reasoning*, pages 2–17. Springer, 2008.

[172] Marc Mezzarobba. *Autour de l'évaluation numérique des fonctions D-finies*. PhD thesis, École polytechnique, 2011.

[173] Ramon E. Moore. *Interval Analysis*. Prentice-Hall, 1966.

[174] Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. *Introduction to interval analysis*, volume 110. Siam, 2009.

[175] Bernard Mourrain. Polynomial-exponential decomposition from moments. *Found. Comput. Math.*, 18(6):1435–1492, 2018.

[176] Jean-Michel Muller. *Elementary Functions, Algorithms and Implementation*. Birkhäuser, Boston, 3rd edition, 2016.

[177] Jean-Michel Muller, Nicolas Brunie, Florent de Dinechin, Claude-Pierre Jeannerod, Mioara Joldes, Vincent Lefèvre, Guillaume Melquiond, Nathalie Revol, and Serge Torres. *Handbook of Floating-Point Arithmetic*. Birkhäuser Boston, 2018.

[178] Mitsuhiro T Nakao. A numerical approach to the proof of existence of solutions for elliptic problems. *Japan Journal of Applied Mathematics*, 5(2):313, 1988.

[179] Mitsuhiro T Nakao. Numerical verification methods for solutions of ordinary and partial differential equations. *Numerical Functional Analysis and Optimization*, 22(3-4):321–356, 2001.

[180] F. Natterer. *The mathematics of computerized tomography*, volume 32 of *Classics in Applied Mathematics*. (SIAM), Philadelphia, PA, 2001.

[181] Markus Neher, Kenneth R Jackson, and Nedialko S Nedialkov. On Taylor model based integration of ODEs. *SIAM J. Numer. Anal.*, 45(1):236–262, 2007.

[182] Yurii Nesterov. Squared functional systems and optimization problems. In *High performance optimization*, pages 405–440. Springer, 2000.

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

[184] A. Neumaier. *The Wrapping Effect, Ellipsoid Arithmetic, Stability and Confidence Regions*, pages 175–190. Springer Vienna, Vienna, 1993.

[185] Arnold Neumaier. Taylor forms—use and limits. *Reliab. Comput.*, 9(1):43–79, 2003.

[186] Octavia Maria Nica-Bolojan. *Fixed point methods for nonlinear differential systems with nonlocal conditions*. PhD thesis, Babes-Bolyai University of Cluj-Napoca, 2013.

[187] Douglas D. Novaes and Joan Torregrosa. On extended Chebyshev systems with positive accuracy. *J. Math. Anal. Appl.*, 448(1):171–186, 2017.

[188] Toshinori Oaku. Algorithms for integrals of holonomic functions over domains defined by polynomial inequalities. *J. Symbolic Comput.*, 50:1–27, 2013.

[189] Shin'ichi Oishi. Numerical verification of existence and inclusion of solutions for nonlinear operator equations. *Journal of Computational and Applied Mathematics*, 60(1-2):171–185, 1995.

[190] Jack Oliver. Rounding error propagation in polynomial evaluation schemes. *J. Comput. Appl. Math.*, 5(2):85–97, 1979.

[191] Sheehan Olver and Alex Townsend. A fast and well-conditioned spectral method. *SIAM Rev.*, 55(3):462–489, 2013.

[192] J. M. Ortega and W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*, volume 30 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000. Reprint of the 1970 original.

[193] 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.

[194] Stefan Paszkowski. *Zastosowania numeryczne wielomianów i szeregów Czebyszewa*. Państwowe Wydawn. Naukowe, 1975. in Polish.

[195] A. I. Perov. On the Cauchy problem for a system of ordinary differential equations. *Približ. Metod. Rešen. Differencial'. Uravnen. Vyp.*, 2:115–134, 1964.

[196] I. G. Petrovskiĭ and E. M. Landis. On the number of limit cycles of the equation $dy/dx = P(x, y)/Q(x, y)$, where P and Q are polynomials of 2nd degree. *Mat. Sb. N.S.*, 37(79):209–250, 1955. Amer. Math. Soc. Transl. **16** (2) (1958), 177–221.

[197] S. Unnikrishna Pillai and A. Papoulis. *Probability, random variables, and stochastic processes*, volume 2. McGraw-Hill, 2002.

[198] Gerlind Plonka and Manfred Tasche. Fast and numerically stable algorithms for discrete cosine transforms. *Linear Algebra Appl.*, 394:309–345, 2005.

[199] Michael Plum. Computer-assisted existence proofs for two-point boundary value problems. *Computing*, 46(1):19–34, 1991.

[200] Michael Plum. Numerical existence proofs and explicit bounds for solutions of nonlinear elliptic boundary value problems. *Computing*, 49(1):25–44, 1992.

[201] Valentina Popescu. *Towards fast and certified multiple-precision librairies*. PhD thesis, Université de Lyon, 2017.

[202] M. J. D. Powell. On the maximum errors of polynomial approximations defined by interpolation and by least squares criteria. *Comput. J.*, 9:404–407, 1967.

[203] Radu Precup. The role of matrices that are convergent to zero in the study of semilinear operator systems. *Math. Comput. Model.*, 49(3):703–708, 2009.

[204] Univalent Foundations Program. *Homotopy type theory: Univalent foundations of mathematics*. Univalent Foundations, 2013. `https://homotopytypetheory.org/book`.

[205] Louis B. Rall. *Computational solution of nonlinear operator equations*. Robert E. Krieger Publishing Co., Inc., Huntington, N.Y., 1979. Corrected reprint of the 1969 original.

[206] Luc Rebillard. *Étude théorique et algorithmique des séries de Chebyshev solutions d'équations différentielles holonomes*. PhD thesis, Institut National Polytechnique de Grenoble-INPG, 1998.

[207] Rembert Reemtsen and Jan-J. Rückmann. *Semi-Infinite Programming*, volume 25. Springer Science & Business Media, 1998.

[208] E. Remes. Sur le calcul effectif des polynômes d'approximation de Tchebichef (in French). *Compt. Rend. Acad. Sci.*, 199:337–340, 1934.

[209] E. Remes. Sur un procédé convergent d'approximations successives pour déterminer les polynômes d'approximation (in French). *Compt. Rend. Acad. Sci.*, 198:2063–2065, 1934.

[210] Nathalie Revol and Fabrice Rouillier. Motivations for an arbitrary precision interval arithmetic and the MPFI library. *Reliab. Comput.*, 11(4):275–290, 2005.

[211] Annalisa Riccardi, Chiara Tardioli, and Massimiliano Vasile. *An intrusive approach to uncertainty propagation in orbital mechanics based on Tchebycheff polynomial algebra*, pages 707–722. Advances in Astronautical Sciences, AAS/AIAA Astrodynamics Specialist Conference, August 9-13, 2015, Vail, Colorado, U.S.A. American Astronautical Society, 8 2015.

[212] Theodore J. Rivlin. *The Chebyshev Polynomials*. Wiley, 1974.

[213] François Robert. *Étude et utilisation de normes vectorielles en analyse numérique linéaire (in French)*. PhD thesis, Université de Grenoble, 1968.

[214] A. Robertson, G. Inalhan, and J. P. How. Formation control strategies for a separated spacecraft interferometer. In *Proceedings of the 1999 American Control Conference*, volume 6, pages 4142–4147, June 1999.

[215] A. Rocca, V. Magron, and T. Dang. Certified Roundoff Error Bounds using Bernstein Expansions and Sparse Krivine-Stengle Representations. In *24th IEEE Symposium on Computer Arithmetic*. IEEE, 2017.

[216] Robert Roussarie. *Bifurcation of planar vector fields and Hilbert's sixteenth problem*, volume 164 of *Progress in Mathematics*. Birkhäuser Verlag, Basel, 1998.

[217] W. Rudin. *Real and complex analysis (3rd)*. New York: McGraw-Hill Inc, 1986.

[218] W. Rudin. *Functional analysis*. McGraw-Hill, 1991.

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

[220] Siegfried M Rump. Intlab—interval laboratory. In *Developments in reliable computing*, pages 77–104. Springer, 1999.

[221] Siegfried M. Rump. Verification methods: rigorous results using floating-point arithmetic. *Acta Numer.*, 19:287–449, 2010.

[222] Bruno Salvy. D-finiteness: Algorithms and applications. In Manuel Kauers, editor, *ISSAC 2005: Proceedings of the 18th International Symposium on Symbolic and Algebraic Computation, Beijing, China, July 24-27, 2005*, pages 2–3. ACM Press, 2005. Abstract for an invited talk.

[223] Bruno Salvy. Linear differential equations as a data-structure. *Found. Comput. Math.*, 2019.

[224] Bruno Salvy and Paul Zimmermann. Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable. *ACM Trans. Math. Software*, 20(2):163–177, 1994.

[225] Laurent Schwartz. *Théorie des distributions*. Publications de l'Institut de Mathématique de l'Université de Strasbourg, No. IX-X. Nouvelle édition, entiérement corrigée, refondue et augmentée. Hermann, Paris, 1966.

[226] Romain Serra, Denis Arzelier, Florent Bréhard, and Mioara Joldes. Fuel-optimal impulsive fixed-time trajectories in the linearized circular restricted 3-body-problem. In *IAF Astrodynamics Symposium in 69TH international astronautical congress (IAC 2018)*, pages 1–9, 2018.

[227] Romain Serra, Denis Arzelier, Mioara Joldes, Jean-Bernard Lasserre, Aude Rondepierre, and Bruno Salvy. Fast and accurate computation of orbital collision probability for short-term encounters. *J. Guidance Control Dynam.*, 39(5):1009–1021, 2016.

[228] Alexander Shapiro. Semi-infinite programming, duality, discretization and optimality conditions. *Optimization*, 58(2):133–161, 2009.

[229] Song Ling Shi. A concrete example of the existence of four limit cycles for plane quadratic systems. *Sci. Sinica*, 23(2):153–158, 1980.

[230] S. K. Shrivastava. Orbital Perturbations and Stationkeeping of Communication Satellites. *Journal of Spacecraft*, 15(2), 1978.

[231] Marcel J. Sidi. *Spacecraft Dynamics and Control*. Cambridge University Press, 1997.

[232] Alexey Solovyev, Marek S. Baranowski, Ian Briggs, Charles Jacobsen, Zvonimir Rakamarić, and Ganesh Gopalakrishnan. Rigorous estimation of floating-point round-off errors with symbolic Taylor expansions. *ACM Trans. Program. Lang. Syst.*, 41(1):2:1–2:39, December 2018.

[233] E. M. Soop. *Handbook of Geostationary Orbits*. Kluwer Academic Publishers Group, 1994.

[234] R. P. Stanley. Differentiably finite power series. *European J. Combin.*, 1(2):175–188, 1980.

[235] G. Strang. The discrete cosine transform. *SIAM Rev.*, 41(1):135–147, 1999.

[236] Adam Wojciech Strzeboński. Computing in the field of complex algebraic numbers. *J. Symbolic Comput.*, 24(6):647–656, 1997.

[237] Joshua Sullivan, Sebastian Grimberg, and Simone D'Amico. Comprehensive survey and assessment of spacecraft relative motion dynamics models. *Journal of Guidance, Control, and Dynamics*, 40(8):1837–1859, 2017.

[238] Teruo Sunaga. Theory of interval algebra and its application to numerical analysis. *RAAG memoirs*, 2(29-46):209, 1958.

[239] G. Szegő. *Orthogonal polynomials*. American Mathematical Society, Providence, R.I., fourth edition, 1975. American Mathematical Society, Colloquium Publications, Vol. XXIII.

[240] Nobuki Takayama. An algorithm of constructing the integral of a module - an infinite dimensional analog of gröbner basis. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation, Tokyo, Japan, 1990*, pages 206–211, 1990.

[241] M. Tillerson, G. Inalhan, and J. P. How. Co-ordination and control of distributed spacecraft systems using convex optimization techniques. *International Journal of Robust and Nonlinear Control*, 12(2-3):207–242, 2002.

[242] Arnaud Tisserand. High-performance hardware operators for polynomial evaluation. *International Journal of High Performance Systems Architecture (IJHPSA)*, 1(1):14–23, 2007.

[243] Lloyd Nicholas Trefethen. *Approximation Theory and Approximation Practice*. SIAM, 2013. See `http://www.chebfun.org/ATAP/`.

[244] J. Tschauner. Elliptic orbit rendezvous. *AIAA Journal*, 5(6):1110–1113, 1967.

[245] J. Tschauner and P. Hempel. Optimale Beschleunigungsprogramme fur das Rendezvous-Manover. *Acta Astronautica*, 10(5-6):296–307, 1964.

[246] Warwick Tucker. A rigorous ODE solver and Smale's 14th problem. *Found. Comput. Math.*, 2(1):53–117, 2002.

[247] Warwick Tucker. *Validated numerics: a short introduction to rigorous computations*. Princeton University Press, 2011.

[248] R. H. Tütüncü, K. C. Toh, and M. J. Todd. Solving semidefinite-quadratic-linear programs using SDPT3. *Math. Program.*, 95(2, Ser. B):189–217, 2003. Computational semidefinite and second order cone programming: the state of the art.

[249] David A. Vallado. *Fundamentals of astrodynamics and applications*. Space Technology Series, 1997.

[250] Jan Bouwe Van Den Berg and Jean-Philippe Lessard. Chaotic braided solutions via rigorous numerics: Chaos in the Swift–Hohenberg equation. *SIAM J. Appl. Dyn. Syst.*, 7(3):988–1031, 2008.

[251] Jan Bouwe van den Berg and Jean-Philippe Lessard. Rigorous numerics in dynamics. *Notices of the AMS*, 62(9), 2015.

[252] J. van der Hoeven. Ball arithmetic. Technical report, HAL, 2009. `http://hal.archives-ouvertes.fr/hal-00432152/fr/`.

[253] Jean Van Heijenoort. *From Frege to Gödel: a source book in mathematical logic, 1879-1931*, volume 9. Harvard University Press, 1967.

[254] A. N. Varchenko. Estimation of the number of zeros of an abelian integral depending on a parameter, and limit cycles. *Funktsional. Anal. i Prilozhen.*, 18(2):14–25, 1984. Functional Anal. Appl. **18** (1984), no. 2, 98–108.

[255] Massimiliano Vasile, Carlos Ortega Absil, and Annalisa Riccardi. Set propagation in dynamical systems with generalised polynomial algebra and its computational complexity. *Communications in Nonlinear Science and Numerical Simulation*, 75:22 – 49, 2019.

[256] J. von Neumann. Zur Einführung der transfiniten Zahlen. *Acta Litt. Sci. Szeged*, 1:199–208, 1923.

[257] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge University Press, Cambridge, third edition, 2013.

[258] S. Wang and P. Yu. Existence of 121 limit cycles in a perturbed planar polynomial Hamiltonian vector field of degree 11. *Chaos Solitons Fractals*, 30(3):606–621, 2006.

[259] G. A. Watson. The calculation of best restricted approximations. *SIAM J. Numer. Anal.*, 11(4):693–699, 1974.

[260] Abdul-Majid Wazwaz. *Linear and nonlinear integral equations: methods and applications*. Springer Science & Business Media, 2011.

[261] Hassler Whitney. *Geometric integration theory*. Princeton University Press, 1957.

[262] Roderick Wong. *Asymptotic approximations of integrals*, volume 34. SIAM, 2001.

[263] Charles E. Woodruff. The evolution of modern numerals from ancient tally marks. *The American Mathematical Monthly*, 16(8/9):125–133, 1909.

[264] Kuan Xu. The Chebyshev points of the first kind. *Appl. Numer. Math.*, 102:17–30, 2016.

[265] Nobito Yamamoto. A numerical verification method for solutions of boundary value problems with local uniqueness by Banach's fixed-point theorem. *SIAM J. Numer. Anal.*, 35(5):2004–2013, 1998.

[266] T Yamamoto. A unified derivation of several error bounds for Newton's process. *J. Comput. Appl. Math.*, 12:179–191, 1985.

[267] Koji Yamanaka and Finn Ankersen. New state transition matrix for relative motion on an arbitrary elliptical orbit. *J. Guidance Control Dynam.*, 25(1):60–66, 2002.

[268] Olivier Zarrouati. *Trajectoires spatiales*. Cépaduès-e edition, 1987.

[269] Doron Zeilberger. A holonomic systems approach to special functions identities. *J. Comput. Appl. Math.*, 32(3):321–368, 1990.

[270] Jens Zemke. b4m: A free interval arithmetic toolbox for MATLAB, 1999.

[271] A. Zygmund. *Trigonometric series. Vol. I, II.* Cambridge Mathematical Library. Cambridge University Press, Cambridge, third edition, 2002.