Autour de l'évaluation numérique des fonctions D-finies

Marc Mezzarobba

École doctorale de l'École polytechnique

Autour de l'évaluation numérique des fonctions D-finies

Thèse

présentée par

Marc Mezzarobba

pour obtenir le grade de Docteur de l'École polytechnique

spécialité : Informatique

soutenue publiquement le 27 octobre 2011

après avis des rapporteurs

Jonathan Borwein,	University of Newcastle (Australie),
Richard Brent,	Australian National University (Australie),
Felix Ulmer,	Université de Rennes

et devant le jury composé de

Jean-Michel Muller,	CNRS & ENS de Lyon,	*président*,
Bruno Salvy,	INRIA,	*directeur de thèse*,
James H. Davenport,	University of Bath (Royaume-Uni),	
François Morain,	École polytechnique,	
Felix Ulmer,	Université de Rennes,	
Joris van der Hoeven,	CNRS & École polytechnique	

Titre. Autour de l'évaluation numérique des fonctions D-finies

Résumé. Les fonctions D-finies (ou holonomes) à une variable sont les solutions d'équations différentielles linéaires à coefficients polynomiaux. En calcul formel, il s'est avéré fructueux depuis une vingtaine d'années d'en développer un traitement algorithmique unifié. Cette thèse s'inscrit dans cette optique, et s'intéresse à l'évaluation numérique des fonctions D-finies ainsi qu'à quelques problèmes apparentés. Elle explore trois grandes directions. La première concerne la *majoration* des coefficients des développements en série de fonctions D-finies. On aboutit à un algorithme de calcul automatique de majorants accompagné d'un résultat de finesse des bornes obtenues. Une seconde direction est la mise en pratique de l'algorithme « bit burst » de Chudnovsky et Chudnovsky pour le *prolongement analytique numérique* à précision arbitraire des fonctions D-finies. Son implémentation est l'occasion de diverses améliorations techniques. Ici comme pour le calcul de bornes, on s'attache par ailleurs à couvrir le cas des points singuliers réguliers des équations différentielles. Enfin, la dernière partie de la thèse développe une méthode pour calculer une *approximation polynomiale* de degré imposé d'une fonction D-finie sur un intervalle, via l'étude des développements en série de Tchebycheff de ces fonctions. Toutes les questions sont abordées avec un triple objectif de rigueur (résultats numériques garantis), de généralité (traiter toute la classe des fonctions D-finies) et d'efficacité. Pratiquement tous les algorithmes étudiés s'accompagnent d'implémentations disponibles publiquement.

Mots-clés : calcul formel, fonctions spéciales, fonctions D-finies, calcul numérique certifié, équations différentielles linéaires, points singuliers réguliers, récurrences linéaires, asymptotique, arithmétique multi-précision, séries majorantes, scindage binaire, *bit burst*, séries de Tchebycheff.

Title. Around the Numerical Evaluation of D-Finite Functions

Abstract. D-Finite functions of one variable (also known as holonomic functions) are the solutions of linear ordinary differential equations with polynomial coefficients. Developing a unified treatment of these functions has proven fruitful in Computer Algebra for the last twenty years. In line with these ideas, the present thesis is concerned with the numeric evaluation of D-finite functions and related issues. We focus on three directions. First, we describe an algorithm to compute *upper bounds* on the coefficients of power series expansions of D-finite functions, starting from the associated differential equations. The main originality of our approach is that we state a tightness criterion with respect to which the computed bounds are optimal. Our second focus is the multiple precision *analytic continuation* of D-finite functions. Here, we put into practice the "bit burst" method of Chudnovsky and Chudnovsky by implementing it, eliminating several redundancies and suggesting various other improvements. Both the bound computation algorithm and the numeric analytic continuation cover the case of generalized series expansions at regular singular points of differential equations. Finally, we develop a method for the approximation of D-finite functions by *polynomials* of prescribed degree over a segment, based on their Chebyshev expansions. All these questions are addressed with the triple goal of mathematical rigor (all approximations include error bounds), generality (the entire class of D-finite functions is treated in a uniform way) and efficiency. In almost all cases, our algorithms are accompanied by publicly available implementations.

Keywords: computer algebra, special functions, D-finite functions, rigorous numerical computation, linear ordinary differential equations, regular singular points, linear recurrences, asymptotics, multiple-precision arithmetics, majorizing series, binary splitting, bit burst, Chebyshev series.

Thèse préparée au sein du projet Algorithms de l'INRIA Paris-Rocquencourt, Domaine de Voluceau, Rocquencourt, 78153 Le Chesnay, ainsi que du centre de recherche commun INRIA-MSR, Parc Orsay Université, 28 rue Jean Rostand, 91893 Orsay Cedex.

Version du 24 janvier 2012 (déposée à la bibliothèque).

Adresse de cette version : `http://marc.mezzarobba.net/these/these-mezzarobba-20120124.pdf`

Adresse de ce document : `http://marc.mezzarobba.net/these/these-mezzarobba.pdf`

Table des matières

Remerciements

« And then there was Abenthy, my first real teacher. He taught me more than all the others set end to end. If not for him, I would never have become the man I am today.

I ask that you not hold it against him. He meant well. »

— Patrick Rothfuss [211, p. 60]

Je voudrais remercier mon directeur de thèse, Bruno Salvy, pour tout ce qu'il m'a appris au cours de ces années, mais aussi pour son dynamisme communicatif, son optimisme inébranlable... et pour avoir su se rendre disponible aux moments cruciaux malgré un emploi du temps chroniquement surchargé ! Merci, avec lui, à Alin Bostan et Frédéric Chyzak, les deux autres permanents de notre sous-équipe. Merci à Mohab Safey El Din, qui m'a fait découvrir le calcul formel au cours d'un stage de fin de licence et donné envie de continuer dans cette voie.

Merci à Jonathan Borwein, Richard Brent et Felix Ulmer d'avoir accepté d'être rapporteurs de cette thèse, parfois malgré la barrière de la langue. Merci à Joris van der Hoeven, dont les travaux ont été une source d'inspiration constante et dont le logiciel TEXMACS m'a servi à préparer ce mémoire. Merci à James Davenport, François Morain et Jean-Michel Muller pour leur présence dans le jury.

Merci à mon « jumeau » de thèse Alexandre Benoit pour tout le temps passé à bosser — ou pas — ensemble. (J'espère que ce n'est pas fini !) Merci à nos cobureaux occasionnels ou de passage, en particulier Shaoshi Chen et Élie de Panafieu, au reste de l'équipe des thésards d'Orsay, dont Cyril Cohen, François Garillot et Jérémy Planul, ainsi qu'à ceux de Palaiseau, notamment Jérémy Berthomieu, Luca De Feo, et Romain Lebreton. Bon courage à tous ceux qui terminent leur thèse en ce moment !

Merci à Sylvain Chevillard et à Mioara Joldeş pour des collaborations aussi plaisantes qu'intéressantes. Merci à Éric Schost, qui m'a accueilli pendant un mois au Canada pour travailler sur la complexité bilinéaire. Merci à Paul Zimmermann pour — entre autres — m'avoir permis de participer à l'écriture du livre *Calcul mathématique avec Sage*, et aux autres auteurs : Alexandre Casamayou, Nathann Cohen, Guillaume Connan, Thierry Dumont, Laurent Fousse, François Maltey, Matthias Meulien, Clément Pernet et Nicolas Thiéry. Merci aussi à Bertrand Meyer, avec qui j'ai collaboré sur la traduction française du « tutoriel » de Sage.

Merci à Alexis Darrasse, Philippe Dumas, Stefan Gerhold, Manuel Kauers, Pierre Nicodème, Nicolas Le Roux, Carine Pivoteau, Flavia Stan, et tous les membres du projet Algorithms ainsi que du centre de recherche commun MSR-Inria pour bien des heures de discussions tantôt fort sérieuses, tantôt beaucoup moins. Merci à Virginie Collette et Martine Thirion ainsi qu'à Élisabeth Delbecq pour leur efficacité face aux ordres de mission et autres dossiers de recrutement.

Merci à mes collègues d'enseignement Philippe Chassignet, Jean-Christophe Filliâtre, Steve Oudot, Yann Ponty, Xavier Rival et David Savourey.

Merci aux nombreux chercheurs avec lesquels j'ai eu l'occasion de discuter au fil des manifestations scientifiques, et en particulier à Cyril Banderier, Moulay Barkatou, Nicolas Brisebarre, Pascal Giorgi, Grégoire Lecerf, Guillaume Moroz, Bernard Mourrain, Adrien Poteaux et Jacques-Arthur Weil. Merci aux *referees* anonymes des articles écrits pendant ma thèse, dont les remarques m'ont permis d'améliorer le con-

tenu de ce mémoire. Merci à Michèle Loday-Richaud et (indirectement) à Anne Duval pour leurs indications sur l'asymptotique des solutions d'équations aux différences.

Merci à mes enseignants au fil des années, notamment à Marc Bonnier, Fernand Martin et Franck Taieb.

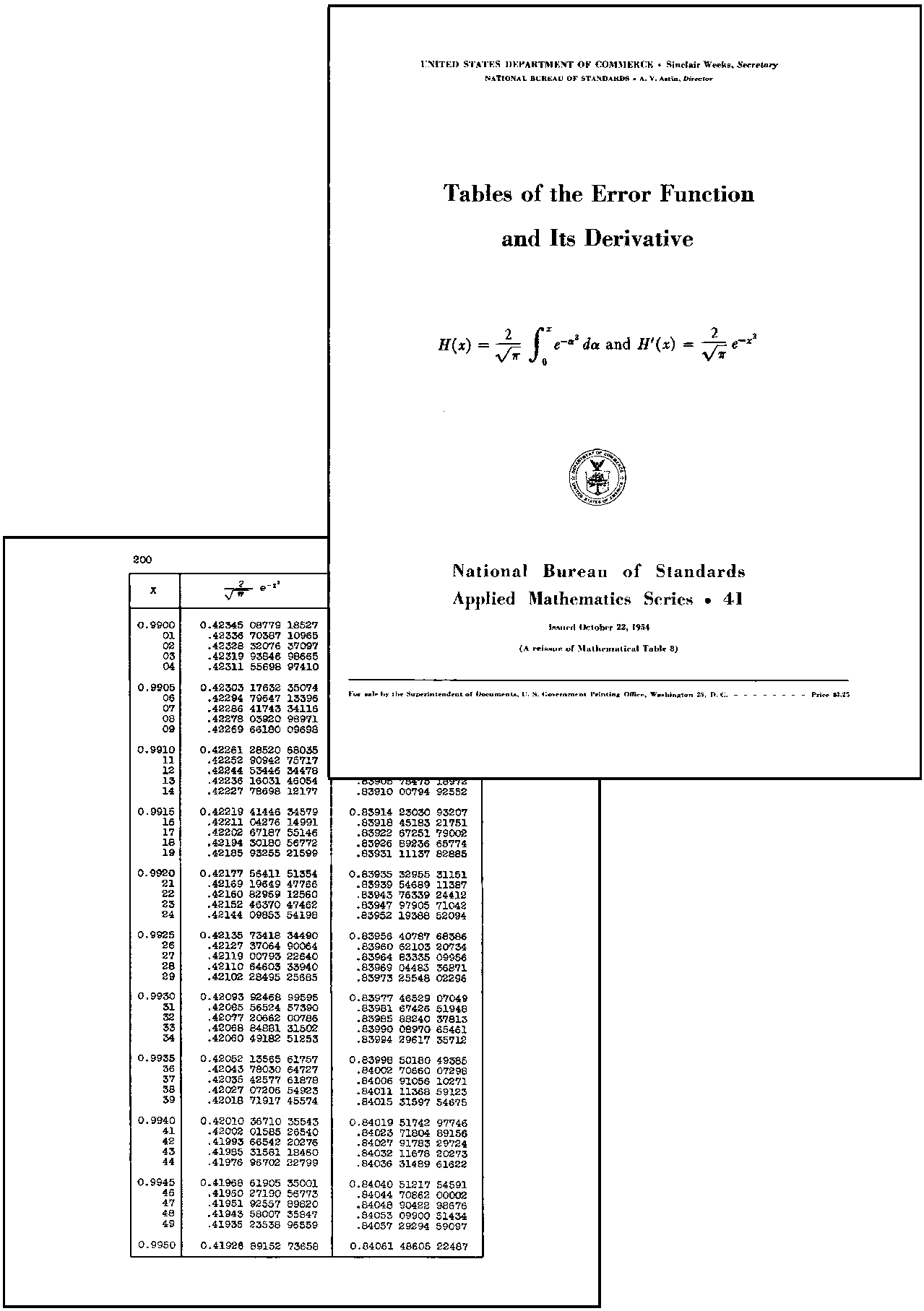
Merci aux familles Moreau-Poullaouec-Vaugon pour leur accueil à une phase critique de la rédaction. Merci à ma famille.

Bîp à Anne.

In memoriam

Sur un registre hélas bien différent, l'environnement scientifique dans lequel j'ai eu la chance de travailler n'existerait pas sans Philippe Flajolet. Le décès brutal de Philippe en mars 2011 a été un choc bien au-delà de son équipe, et je voudrais saluer sa mémoire.

J'ai une pensée enfin pour mes camarades de prépa puis d'école Frédéric Bouchier, Thibaut Kirchner et Jean Lorenzi.



UNITED STATES DEPARTMENT OF COMMERCE • Sinclair Weeks, *Secretary*
NATIONAL BUREAU OF STANDARDS • A. V. Astin, *Director*

Tables of the Error Function and Its Derivative

$$H(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-\alpha^2}\, d\alpha \text{ and } H'(x) = \frac{2}{\sqrt{\pi}} e^{-x^2}$$

National Bureau of Standards
Applied Mathematics Series • 41

Issued October 22, 1954

(A reissue of Mathematical Table 8)

For sale by the Superintendent of Documents, U. S. Government Printing Office, Washington 25, D. C. - - - - - - - - - Price $3.25

200

x	$\frac{2}{\sqrt{\pi}} e^{-x^2}$	
0.9900	0.42345 08779 18527	
01	.42336 70367 10965	
02	.42328 32076 37097	
03	.42319 93846 98665	
04	.42311 55698 97410	
0.9905	0.42303 17632 35074	
06	.42294 79647 13396	
07	.42286 41743 34116	
08	.42278 03920 98971	
09	.42269 66180 09698	
0.9910	0.42261 28520 68035	
11	.42252 90942 75717	
12	.42244 53446 34478	
13	.42236 16031 46054	
14	.42227 78698 12177	.83908 78478 18972
		.83910 00794 92552
0.9915	0.42219 41446 34579	0.83914 23030 93207
16	.42211 04276 14991	.83918 45183 21751
17	.42202 67187 55146	.83922 67251 73002
18	.42194 30180 56772	.83926 89236 65774
19	.42185 93255 21598	.83931 11137 82885
0.9920	0.42177 56411 51354	0.83935 32955 31151
21	.42169 19649 47766	.83939 54689 11387
22	.42160 82969 12560	.83943 76339 24412
23	.42152 46370 47462	.83947 97905 71042
24	.42144 09853 54198	.83952 19388 52094
0.9925	0.42135 73418 34490	0.83956 40787 66586
26	.42127 37064 90064	.83960 62103 20734
27	.42119 00793 22540	.83964 83335 09956
28	.42110 64603 33940	.83969 04483 36871
29	.42102 28495 25685	.83973 25548 02296
0.9930	0.42093 92468 99595	0.83977 46529 07049
31	.42085 56524 57390	.83981 67426 51948
32	.42077 20662 00786	.83985 88240 37813
33	.42068 84881 31502	.83990 08970 65461
34	.42060 49182 51253	.83994 29617 35712
0.9935	0.42052 13565 61757	0.83998 50180 49385
36	.42043 78030 64727	.84002 70660 07298
37	.42035 42577 61878	.84006 91056 10271
38	.42027 07206 54923	.84011 11368 59123
39	.42018 71917 45574	.84015 31597 54675
0.9940	0.42010 36710 35543	0.84019 51742 97746
41	.42002 01585 26540	.84023 71804 89156
42	.41993 66542 20276	.84027 91783 29724
43	.41985 31581 18460	.84032 11678 20273
44	.41976 96702 22799	.84036 31489 61622
0.9945	0.41968 61905 35001	0.84040 51217 54591
46	.41960 27190 56773	.84044 70862 00002
47	.41951 92557 89820	.84048 90422 98676
48	.41943 58007 35847	.84053 09900 51434
49	.41935 23538 96559	.84057 29294 59097
0.9950	0.41926 89152 73658	0.84061 48605 22487

Figure 1. Pages extraites des tables de la fonction d'erreur préparées par le *Mathematical Tables Project* [68].

Introduction

> « Let us consider a function which is defined by a differential equation (or even purely algebraic equation) whose coefficients are *rational functions of* x. [...] In fact, the majority of the important functions encountered in the advanced chapters of physics and engineering belong to this category, if we add those functions which are not *directly* defined by such a law but which can be *conceived* as the solution of such differential equation. »
>
> — Cornelius Lánczos, *Applied Analysis* (1956) [154, p. 464]

1 Des tables et des programmes

Plus personne ne consulte les tables numériques du *Mathematical Tables Project*, 28 volumes particulièrement détaillés préparés entre 1938 et 1948 par une armée de calculateurs humains, dans le cadre d'un programme issu du *New Deal* [118]. Scientifiques et ingénieurs n'ont pourtant pas cessé d'avoir recours à des valeurs numériques de plus en plus précises des fonctions mathématiques les plus variées !

Les tables de logarithmes ont cédé la place aux calculatrices de poche voici déjà quelques décennies. Quant à l'utilisateur en quête de valeurs d'une fonction spéciale comme la fonction d'erreur,

$$\operatorname{erf} x = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \, \mathrm{d}t,$$

c'est vers son ordinateur qu'il se tournera naturellement. La ligne de commande interactive de logiciels de calcul formel comme Maple [166] ou Sage [225]

```
> evalf(erf(0.9947));
```

```
.8404890423
```

```
sage: erf(0.9947)
```

```
0.840489042298676
```

remplace avantageusement les tables de fonctions spéciales. D'autres ouvrages de référence, les tables de sommes et d'intégrales par exemple, connaissent progressivement le même sort.

Les 53 bits significatifs des flottants « double précision », standard de fait du calcul numérique sur ordinateur, correspondent peu ou prou aux 15 chiffres décimaux de la table reproduite ci-contre. De plus en plus, les langages de programmation comportent une poignée de fonctions spéciales, implémentées à ce niveau de précision. La fonction d'erreur fait ainsi partie de la bibliothèque mathématique standard du langage C depuis la norme C99 [133], rendant directement accessible au programmeur l'équivalent des volumes de tables correspondants :

```
$ gcc erf.c -lm && ./a.out   # voir code en figure 2 page 13
```

```
0.840489042298676
```

Les routines de ce genre s'appuient en général sur des approximations polynomiales précalculées : ce sont en quelque sorte des tables « optimisées » [185, 186].

Une procédure d'évaluation qui cible un domaine et une précision non bornés doit, elle, coder l'équivalent du processus de construction de la table. La précision arbitraire est le domaine de bibliothèques spécialisées comme MPFR [183] :

```
$ gcc erf-mpfr.c -lmpfr && ./a.out   # voir code en figure 2
  8.40489042298676266022069152431166319144369176412277157803699293926472822077680902481423795 64e-01
```

ainsi, à nouveau, que des logiciels de calcul formel :

```
> evalf[80](erf(0.9947));
   .84048904229867626602206915243116631914436917641227715780369929392647282207768090
```

Ceux-ci ont aussi tendance à couvrir des classes de fonctions plus vastes que les seules fonctions élémentaires et fonctions spéciales usuelles. Les fonctions encore un peu plus rares nécessitent du code spécifique.

Si une table numérique est par la force des choses consacrée à une fonction figée, la production de ces tables reposait en son temps sur un nombre somme toute limité de méthodes plus ou moins générales. Ainsi, des techniques comme la méthode tau de Lánczos [154] ou la méthode de Clenshaw [66][1] ont comme champ d'application naturel les solutions d'équations différentielles linéaires à coefficients polynomiaux. Bon nombre de fonctions spéciales sont solutions d'équations de ce type, ne serait-ce qu'en raison de leur origine physique, d'où l'intérêt de passer par là pour les tabuler.

Il en va de même des outils informatiques d'évaluation numérique, à précision machine comme à précision arbitraire. Les implémentations sont presque toujours développées fonction par fonction, en utilisant judicieusement des formules d'approximation choisies au cas par cas, tandis que certaines des *méthodes* employées pour établir puis exploiter ces formules sont applicables à de larges classes de fonctions. Les développeurs s'efforcent bien sûr de partager du code entre les implémentations. Dans le domaine de la précision arbitraire, une stratégie populaire consiste à exprimer les fonctions usuelles en termes des fonctions hypergéométriques généralisées. Ainsi pourra-t-on dans bien des cas calculer $\operatorname{erf} x$ en appliquant plus ou moins directement la formule

$$\operatorname{erf} x = \frac{2\,x}{\sqrt{\pi}}\,{}_1\mathrm{F}_1\left(\begin{matrix}1/2\\3/2\end{matrix}\middle|\,-x^2\right).$$

(C'est ce qu'il se passe dans le deuxième exemple en Maple ci-dessus.) Reste qu'en-dehors des variantes triviales, chaque nouvelle fonction requiert l'attention du programmeur et comporte son lot de bugs.

Grâce aux outils offerts par le calcul formel, il est tentant de pousser l'automatisation un cran plus loin, et de faire des méthodes d'évaluation à adapter au cas de chaque fonction des *algorithmes* applicables uniformément à des classes de problèmes. On unifie et simplifie ainsi le traitement des fonctions classiques tout en permettant celui de fonctions moins classiques. Au lieu de « la fonction d'erreur », on voudrait pouvoir évaluer, par exemple, « la fonction y telle que $y''(z)+2\,z\,y'(z)=0$, $y(0)=0$, et $y'(0)=2/\sqrt{\pi}$ », et ce, en perdant le moins possible des performances et des garanties d'une routine d'évaluation spécialisée :

```
> evaldiffeq({diff(y(z),z,z) + 2*z*diff(y(z),z) = 0, y(0) = 0, D(y)(0) = 2/sqrt(Pi)},
  y(z), 0.9947, 80);
   .84048904229867626602206915243116631914436917641227715780369929392647282207768090
```

C'est là le sujet de ce mémoire.

1. Toutes deux présentées dans le dernier chapitre de ce mémoire.

```c
/* erf.c */
#include <math.h>
#include <stdio.h>
int main(void) {
    printf("%.15g˙", erf(0.9947));
    return 0;
}
```

```c
/* erf-mpfr.c */
#include <stdio.h>
#include <mpfr.h>
int main(void) {
    mpfr_t x, res;
    mpfr_set_default_prec(300);
    mpfr_init_set_str(x, "0.9947", 10,
                      MPFR_RNDN);
    mpfr_init(res);
    mpfr_erf(res, x, MPFR_RNDN);
    mpfr_printf("%Re\n", res);
    return 0;
}
```

Figure 2. Évaluation de la fonction d'erreur depuis un programme en C.

2 Résoudre ou ne pas résoudre

Face à une intégrale, une somme, une équation différentielle, on a souvent le réflexe de *calculer*, de *résoudre*, afin d'arriver à une expression en termes de fonctions connues de la solution ou de l'objet dénoté par la formule. Les formes explicites sont précieuses, et des algorithmes puissants existent pour les trouver. Tout utilisateur du calcul formel a vu son logiciel favori calculer comme par magie une intégrale peu commode.

Sans doute a-t-il aussi été confronté un jour à une réponse « explicite » en termes de fonctions G de Meijer, ou d'autres fonctions spéciales ésotériques. Car il en va des calculs sur ordinateur comme de ceux à la main : afin d'augmenter le pouvoir d'expression des « formules », on est conduit à introduire peu à peu de nouvelles fonctions. Définies comme solutions « canoniques » de telle ou telle équation remarquable, elles s'accompagnent de règles de calcul qui permettent de travailler avec les expressions les faisant intervenir. Le support d'une large gamme de fonctions spéciales est un enjeu conséquent pour un système de calcul formel lorsqu'il est la clé pour représenter sur l'ordinateur les fonctions (et les nombres [30]) que l'on souhaite manipuler.

Une approche un peu différente consiste à se donner les moyens de *calculer avec* ces fonctions même quand elles ne sont pas représentables explicitement. La nuance avec l'adjonction de fonctions spéciales très générales n'est souvent qu'une question de langage. Mais celui des équations est parfois fort naturel du point de vue algorithmique, par exemple s'il s'agit de manipuler les nombres algébriques à travers leurs polynômes annulateurs. Il sous-entend en tout cas un point de vue moins orienté vers les règles de calcul *ad hoc*, et plus vers le traitement uniforme d'une classe d'objets bien définie. La détection des cas particuliers où existent des expressions plus explicites et la recherche de ces dernières est dévolue à une phase de résolution clairement séparée.

Depuis une vingtaine d'années, Zeilberger [264] et ses successeurs ont ainsi montré que des systèmes d'*équations fonctionnelles linéaires* constituent une « bonne » représentation pour des objets comme les suites d'entiers ou les fonctions spéciales. Le premier succès majeur de cette théorie est une méthode algorithmique de *creative telescoping* [264, 265, 255], qui permet de réaliser l'opération de sommation ou d'intégration définie, comme dans

$$\sum_k \binom{n}{k} = 2^n$$

ou encore, I, J, K, Y désignant les fonctions de Bessel,

$$\int_0^\infty x\, J_1(a\,x)\, I_1(a\,x)\, Y_0(x)\, K_0(x)\, \mathrm{d}x = -\frac{\ln(1-a^4)}{2\,\pi\,a^2} \qquad (0<a<1).$$

Ce second exemple est emprunté à la thèse de Chyzak [59], qui étend l'approche de Zeilberger à des systèmes pouvant mêler équations différentielles, récurrences, et d'autres relations fonctionnelles. Je renvoie à celle de Koutschan [149] pour un panorama plus complet et un aperçu de quelques développements récents.

Nous nous concentrons dans ce mémoire sur les plus simples de ces représentations implicites, celles en une seule variable. Précisément, les *fonctions D-finies* du titre (dites aussi holonomes) sont celles-là même que manipulaient déjà Lánczos ou Clenshaw : les *solutions d'équations différentielles linéaires à coefficients polynomiaux*

$$a_r(z)\, y^{(r)}(z) + a_{r-1}(z)\, y^{(r-1)}(z) + \cdots + a_0(z)\, y(z) = 0, \qquad a_k \in \mathbb{C}[z].$$

Nous nous intéressons parallèlement à leurs proches cousines, les *suites P-récursives* (ou holonomes, P-finies, voire D-finies) qui sont les solutions de *récurrences* linéaires à coefficients polynomiaux

$$b_s(n)\, u_{n+s} + b_{s-1}(n)\, u_{n+s-1} + \cdots + b_0(n)\, u_n = 0, \qquad b_k \in \mathbb{C}[n].$$

Tout comme les systèmes fonctionnels plus généraux mentionnés au paragraphe précédent, une telle équation accompagnée de conditions initiales convenables fournit une *représentation exacte* de sa solution qui se prête aux manipulations algorithmiques.

Comme nous l'avons vu, nombre de fonctions élémentaires et de fonctions spéciales sont D-finies. C'est le cas par exemple des sinus, cosinus et arctangente circulaires et hyperboliques, des fonctions d'Airy, des fonctions de Bessel, ou encore de toute la classe des fonctions hypergéométriques classiques et généralisées. La combinatoire est une autre source abondante de fonctions D-finies arbitraires. De nombreuses structures combinatoires naturelles ont des séries génératrices D-finies. De plus, on dispose de méthodes puissantes pour passer d'une description combinatoire récursive d'une famille d'objets à un système d'équations qui les « compte », puis extraire de celui-ci des informations asymptotiques précises, même en l'absence de formule simple pour le nombre d'objets [94, 223]. Enfin, les résultats de calculs symboliques menés *via* l'approche de Zeilberger sont souvent des fonctions ou des suites D-finies en une variable — à plus forte raison si l'on se propose de les évaluer, d'éventuels paramètres ayant pu être spécialisés au préalable.

On pourrait faire remonter l'étude des fonctions D-finies à la seconde moitié du dix-neuvième siècle. Une bonne partie des résultats mathématiques auxquels nous ferons appel datent en effet des travaux de Cauchy, Riemann, Fuchs, Frobenius et d'autres mathématiciens sur les équations différentielles linéaires à coefficients analytiques. Mais c'est le début des années 1980 qui voit se dégager le concept de série D-finie, d'abord en combinatoire, avec un article de Stanley [222] récapitulant les propriétés de base qui en font une classe d'objets remarquables.

Des travaux de Zeilberger naissent ensuite, vers 1990, les premiers algorithmes de calcul formel majeurs utilisant véritablement les *récurrences et équations différentielles linéaires comme structure de données*. Plusieurs implémentations indépendantes de son algorithme de sommation hypergéométrique apparaissent rapidement. Peu après, la première version du module Maple gfun [214] propose une boîte à outils complète de calcul avec comme objets de base ces équations. D'autres outils du même genre suivent [147, 165]. À l'extrême, on peut envisager de remplacer complètement la définition « explicite » dans un système de calcul formel des fonctions qui s'y prêtent par une description « D-finie ». C'est ce que font, à une échelle réduite, l'*Encyclopedia of Special Functions* [172], et à sa suite le projet DDMF présenté au chapitre 2 de ce mémoire.

3 Et l'évaluation numérique ?

Les travaux mentionnés dans la section précédente ont en commun de traiter les séries D-finies comme des séries *formelles* plutôt que comme des fonctions analytiques.

On attend pourtant d'un logiciel de calcul formel qu'il soit capable de tracer les graphes des fonctions qu'on y manipule, ou encore de les évaluer à précision arbitraire. Lorsque l'on traite les fonctions élémentaires et fonctions spéciales une à une, chacune

équations polynomiales	équations différentielles linéaires	récurrences linéaires
nombres algébriques	fonctions D-finies	suites P-récursives
polynôme annulateur	équation différentielle	récurrence
solutions par radicaux	solutions élémentaires, liouvilliennes	ΠΣ-solutions
théorie de Galois	th. de Galois différentielle	th. de Galois aux différences
calcul (certifié) de racines	**évaluation numérique (garantie)**	n-ième terme
...	...	...

Figure 3. Calcul avec des objets implicites : il peut être éclairant de comparer la situation des fonctions D-finies avec celle des nombres algébriques, que l'on manipule naturellement à travers leur polynôme minimal. (Ce parallèle est développé plus avant dans l'introduction de la thèse de Chyzak [59].)

vient habituellement avec non seulement les règles algébriques nécessaires aux manipulations symboliques, mais aussi du code d'évaluation numérique.

Afin de manipuler les fonctions D-finies dans l'optique de la section précédente, il est souhaitable, de même, que l'évaluation numérique fasse partie des opérations disponibles en utilisant les équations différentielles comme structure de donnée. Ce que les systèmes existants offrent en général de plus proche est un intégrateur numérique générique d'équations différentielles. Ces intégrateurs sont assez éloignés des caractéristiques des routines d'évaluation numérique des fonctions usuelles : ils ne donnent des résultats corrects que sur un voisinage limité du point où sont fournies les conditions initiales, n'offrent le plus souvent aucune garantie, ciblent la petite précision plutôt que la précision arbitraire, et finalement ne tirent que très peu parti des propriétés fortes des fonctions D-finies.

Plus généralement, les opérations de nature analytique — « majorer, minorer, approcher », suivant les mots de Dieudonné [76, p. 15] — sur les fonctions D-finies sont comparativement peu développées, que ce soit en théorie ou en pratique. Comme travaux allant explicitement dans cette direction, je ne connais guère que ceux de van der Hoeven [235, 236, 239] sur l'évaluation à grande précision des fonctions D-finies, qui s'inscrivent dans un programme plus général d'analyse complexe effective [240, 243], et ceux de Gerhold, Kauers et Pillwein [109, 141, 143] sur la preuve d'inégalités. Quelques développements plus anciens, notamment une série d'articles des frères Chudnovsky [54, 57] précurseurs de ceux de van der Hoeven déjà cités, répondaient déjà aux mêmes questions du point de vue du calcul formel. En revanche, toute une littérature qui va des mathématiques pures à l'arithmétique d'intervalles en passant par l'analyse numérique (et les méthodes de Lánczos ou de Clenshaw mentionnées plus haut) fourmille d'idées qui mériteraient d'être revisitées pour nos besoins.

Cette thèse contribue à munir la structure de données « fonction D-finie » d'opérations « analytiques », à commencer par l'évaluation numérique. Le leitmotiv est de proposer des solutions, par ordre approximatif d'importance :

Automatiques. Je m'efforce de donner des *algorithmes généraux*, plutôt que des méthodes qui demandent des ajustements manuels pour s'appliquer efficacement à un problème donné. Ces algorithmes sont autant que possible accompagnés d'*implémentations*, qui elles-mêmes fonctionnent pour toute la classe des fonctions D-finies et à précision arbitraire.

Garanties. À l'image des tables mentionnées plus haut, les outils développés visent à pouvoir servir de référence et se doivent donc de fournir des valeurs numériques fiables. Tous les résultats ou presque s'accompagnent donc de *bornes d'erreur rigoureuses*.

Rapides. L'arrivée de machines de bureau puissantes et d'implémentations extrêmement rapides d'algorithmes avancés en arithmétique multiprécision et en calcul formel renforce l'importance pratique de la complexité des algorithmes. En ce qui concerne l'évaluation ou l'approximation de fonctions, la taille de l'expression à évaluer est en règle générale plus petite que celle du résultat. On vise alors une *complexité quasi-optimale*, linéaire en la taille de la sortie à des facteurs logarithmiques près.

Presque tous les algorithmes en jeu sont naturellement *symboliques-numériques*. On appelle ainsi les procédés mêlant calculs exacts et approchés, qu'il s'agisse de passer par des calculs numériques pour obtenir plus rapidement un résultat symbolique, ou inversement de résoudre numériquement un problème mal conditionné grâce à un prétraitement ou des étapes exactes.

Les résultats « analytiques » que nous nous proposons de calculer ne se limitent pas aux valeurs numériques. En particulier, des approximants polynomiaux, ou encore diverses sortes de bornes, sont tout à la fois des intermédiaires pour le calcul de valeurs numériques et des résultats que l'on peut mettre sous une forme intelligible et renvoyer à l'utilisateur. Or, parmi les utilisateurs potentiels d'un outil d'évaluation numérique garanti, se trouvent les développeurs de bibliothèques numériques. Les outils de calcul formel auxquels font appel les algorithmes d'évaluation à la fois automatiques et relativement généraux rejoignent parfois ceux dont eux auraient besoin. Un second axe se dessine : commencer ainsi à développer, toujours concernant la classe des fonctions D-finies, des outils de calcul formel « pour l'arithmétique des ordinateurs », complémentaires d'outils spécialisés comme Sollya [52].

4 Organisation du manuscrit et contributions

Les principales contributions de cette thèse sont

- une méthode de calcul de bornes supérieures fines sur les suites P-récursives et les fonctions D-finies (en collaboration avec B. Salvy) ;
- la mise en pratique, avec plusieurs améliorations, de l'algorithme général de prolongement analytique numérique de Chudnovsky et Chudnovsky ;
- et une méthode de calcul d'approximations polynomiales fines de fonctions D-finies sur un intervalle (en collaboration avec A. Benoit et M. Joldeş).

Ces travaux ont donné lieu à deux publications respectivement dans une conférence et une revue internationales avec comité de lecture,

[174] M. Mezzarobba. NumGfun: a package for numerical and analytic computation with D-finite functions. In *ISSAC'10* [148], pages 139–146.
`http://arxiv.org/abs/1002.3077`

[175] M. Mezzarobba et B. Salvy. Effective bounds for P-recursive sequences. *Journal of Symbolic Computation*, 45(10):1075–1096, 2010.
`http://arxiv.org/abs/0904.2452`

Un travail collectif auquel j'ai participé a par ailleurs fait l'objet un article invité à une conférence internationale :

[19] A. Benoit, F. Chyzak, A. Darrasse, S. Gerhold, M. Mezzarobba et B. Salvy. The dynamic dictionary of mathematical functions (DDMF). In *Mathematical Software – ICMS 2010* [103], pages 35–41.

Un quatrième article,

[20] A. Benoit, M. Joldeş et M. Mezzarobba. Rigorous uniform approximation of D-finite functions using Chebyshev expansions. En préparation.

devrait être soumis prochainement pour publication.

Le mémoire s'organise en une dizaine de chapitres, que l'on peut grouper en cinq parties comme suit. Une partie du contenu est inédite ; outre l'adaptation de l'article en préparation [20], il s'agit pour l'essentiel d'extensions des résultats de [174, 175].

Développements logiciels [174, 19]. Un souci important a été de développer des algorithmes *utilisables* et de les mettre en pratique. Aussi le corps du mémoire s'ouvre-t-il sur deux chapitres de présentation de logiciels développés au cours de ma thèse. C'est l'occasion de compléter cette introduction en illustrant concrètement les problèmes et les résultats étudiés plus en détail par la suite. Le chapitre 1 décrit NumGfun, un module Maple qui regroupe mes implémentations d'une bonne partie des algorithmes de calcul de bornes et d'évaluation numérique étudiés. Il s'agit surtout de donner des exemples de ses possibilités ; toutes les explications détaillées de fonctionnalités spécifiques sont repoussées à la suite. Le chapitre suivant présente le

Dynamic Dictionary of Mathematical Functions, un site web interactif, inspiré du *Handbook of Mathematical Functions* d'Abramowitz et Stegun [7], et qui se veut une vitrine de l'approche par la D-finitude des fonctions spéciales. Les besoins du DDMF ont souvent guidé le développement de NumGfun et plus largement les questions abordées dans cette thèse.

Préliminaires [174]. Suivent quelques rappels mathématiques. Dans le chapitre 3, j'introduis une poignée de conventions et notations, et je rappelle les propriétés des équations différentielles et récurrences linéaires tenues pour acquises par la suite. Le chapitre 4 est plus particulièrement consacré aux points singuliers des équations différentielles linéaires à coefficients analytiques. À nouveau constitué en grande partie de rappels, il contient une étude bibliographique relativement détaillée des variantes de la *méthode de Frobenius*, ainsi qu'une formulation apparemment nouvelle de l'une d'entre elles que je pense mieux adaptée à nos besoins de calcul formel.

Bornes [175, 174]. La partie suivante porte sur le calcul de *bornes supérieures* sur les suites et séries données par des récurrences ou des équations différentielles. Effectuer de façon fiable des opérations de nature analytique sur les séries D-finies exige de contrôler aussi finement que possible leur vitesse de convergence ou encore les valeurs qu'elles peuvent prendre. Avec mon directeur de thèse Bruno Salvy, nous avons proposé un algorithme de calcul de bornes sur les suites P-récursives qui constitue un premier pas dans ce sens. L'originalité de notre approche réside dans l'existence d'un critère de *finesse* auquel répondent les bornes calculées : en l'occurrence, leur comportement asymptotique « ne s'éloigne pas trop » de celui des suites qu'elles majorent. Cet algorithme fait l'objet du chapitre 5. Le chapitre 6 poursuit les variations sur ces bornes en vue de leur application à l'évaluation numérique, notamment en majorant finement les *restes* de développements de Taylor de fonctions D-finies.

Évaluation ponctuelle à grande précision [174]. Les chapitres 7 et 8 entament l'étude de l'évaluation numérique proprement dite. Ils concernent l'évaluation des fonctions (et, anecdotiquement, des suites) D-finies en un point et à grande précision, dans un contexte assez général qui couvre le *prolongement analytique numérique* des solutions d'une équation différentielle et la *connexion* aux points singuliers réguliers. Dans le chapitre 7, je rappelle quelques résultats classiques sur la complexité algorithmique des opérations de base sur les polynômes et les séries. Le sujet central est la construction d'un *arbre de produits* de matrices à coefficients rationnels, et avec lui l'algorithme de calcul du n-ième terme d'une suite récurrente qui s'en déduit. L'analyse de leur complexité dans les cas qui nous intéressent est un peu plus précise que ce que j'ai trouvé dans la littérature. Le chapitre 8 est consacré à l'algorithme d'évaluation général de Chudnovsky et Chudnovsky, fondé sur le calcul d'arbres de produits, ainsi qu'à certaines de ses extensions par van der Hoeven. Je propose une poignée d'améliorations à son efficacité pratique et à l'analyse de sa complexité. Plus généralement, je détaille les ajustements qui font passer de l'algorithme théorique décrit dans les articles originaux à la version implémentée dans NumGfun, et notamment le suivi des erreurs tout au long du calcul, sur la base des bornes établies dans les chapitres précédents.

Approximation uniforme sur un segment [20]. Nous passons pour finir de la grande précision à la précision modérée, et de l'évaluation en un point au calcul d'*approximations polynomiales*. Le dernier chapitre reprend une version préliminaire d'un travail en cours avec deux autres doctorants, Alexandre Benoit et Mioara Joldeş. Nous étudions tout d'abord les *développements en série de Tchebycheff* des fonctions D-finies, et en particulier les récurrences que satisfont leurs coefficients. Sur cette base, nous montrons comment calculer numériquement une approximation de degré donné d'une fonction D-finie sur un segment avec (sous des hypothèses raisonnables) une erreur en norme uniforme proche de l'optimum. Enfin, nous donnons un algorithme pour calculer rigoureusement une *borne*, à nouveau proche de l'optimum, sur la diffé-

rence entre un polynôme comme celui fourni par le premier algorithme et la fonction qu'il approche. Les deux algorithmes ont une complexité arithmétique *linéaire* en le degré du polynôme.

La correspondance entre articles publiés ou en préparation et chapitres du manuscrit, indiquée brièvement ici, est décrite plus en détail au début de chaque chapitre. En résumé, l'article sur le calcul de bornes [175] est repris avec quelques améliorations et étendu ; l'article de présentation de NumGfun [174], qui contient un résumé de résultats relatifs à l'évaluation numérique à grande précision, est complètement réorganisé et son contenu dispersé dans tout le manuscrit ; celui de présentation du DDMF [19] est adapté au contexte de cette thèse ; et celui en préparation sur l'approximation uniforme à base de séries de Tchebycheff [20] est pratiquement recopié en l'état.

Chapitre 1

NumGfun

« Jetzt, wo Miss Maple es sagte, lag es vor ihnen,
klar wie eine saubere Pfütze. »[1]

— Leonie Swann [229, S. 147]

Cette thèse s'accompagne d'un module Maple appelé NumGfun, qui regroupe des implémentations d'une grande partie des algorithmes étudiés ou développés. Dans ce premier chapitre, je donne un aperçu général de NumGfun et j'illustre ses fonctionnalités. C'est en même temps l'occasion d'introduire de manière concrète les problèmes qui nous occuperont dans la suite du mémoire.

J'ai présenté NumGfun dans un article publié à ISSAC 2010 [174], dont le contenu repris et développé ici se trouve réparti entre ce chapitre et plusieurs des suivants. De nombreux exemples sont par ailleurs adaptés d'exposés donnés au cours de ma thèse.

1.1 Présentation

Objet. Le calcul avec les fonctions D-finies et suites P-récursives, suivant la philosophie exposée en introduction, est devenu une fonctionnalité répandue dans les systèmes de calcul formel. Les versions récentes de Mathematica [259] disposent par exemple nativement d'une structure de données appelée `DifferentialRoot`, qui permet de représenter des fonctions D-finies arbitraires et de calculer avec. Son analogue en Maple [166], `DESol`, est plus limité.

Maple vient en revanche depuis longtemps avec le module gfun[2] de Salvy et Zimmermann [214, 110], implémentation pionnière de cette approche dont la première version date de 1992. Des bibliothèques aux fonctionnalités analogues existent dans d'autres systèmes, avec en Mathematica les modules GeneratingFunctions (Mallinger [165]) et SpecialFunctions (Koepf [147]), et des implémentations partielles dans les systèmes libres Mathemagix (van der Hoeven [243, 242]) et FriCAS (Rubey). Sans offrir de boîte à outils complète « à la gfun », bien d'autres systèmes de calcul symbolique disposent au moins de fonctions de base de manipulation de récurrences ainsi que des algorithmes de sommation hypergéométrique de Gosper et de Zeilberger.

À l'exception peut-être du module holonomix de Mathemagix, ces logiciels ciblent essentiellement les applications des séries D-finies en combinatoire et la preuve d'identités entre expressions D-finies. Pour cette raison, les solutions d'équations différentielles auxquelles s'appliquent les opérations sont les solutions séries formelles plutôt qu'analytiques. Les solutions au voisinage de singularités qui ne s'expriment pas comme des séries entières ne sont pas non plus prises en compte. Il existe cependant par ailleurs de puissants outils de calcul formel dédiés à l'étude locale des solutions d'équations différentielles, et gfun interagit fort bien, par exemple, avec le module DEtools de Maple.

1. « Miss Maple war das klügste Schaf von ganz Glennkill. Manche behaupteten sogar, sie sei das klügste Schaf der Welt. Doch niemand konnte das nachweisen. » [229, S. 12]

2. La version distribuée avec Maple (gfun 3.20 avec Maple 15) est relativement ancienne. Nous utilisons dans ce mémoire une version plus récente (3.50 ou ultérieure), téléchargeable depuis les pages web du projet Algorithms [205].

Le logiciel présenté ici, NumGfun, est une extension à gfun dédiée à *l'évaluation* des suites ou fonctions définies par les équations traitées, et plus généralement à leur manipulation « analytique ». Il offre, entre autres, un moteur de *prolongement analytique numérique* garanti de solutions d'équations différentielles linéaires à coefficients polynomiaux, et des fonctions de calcul de *bornes* symboliques sur les valeurs de suites récurrentes.

Distribution. NumGfun a évolué à partir d'une première version (presque entièrement réécrite entre-temps) qui date de mon stage de M2. Celle utilisée pour l'instant dans ce document est une version de développement qui doit déboucher prochainement sur la version stable accompagnant la forme finale de cette thèse. Après quelques versions indépendantes à ses débuts, NumGfun est désormais un sous-module de gfun. Il est distribué avec lui au sein de la bibliothèque Algolib [205], sous les conditions de la *GNU Lesser General Public Licence*, version 2.1 [99].

Travaux apparentés. Les algorithmes mis en œuvre dans NumGfun trouvent en majorité leur origine dans les travaux des frères Chudnovsky [53, 54, 55, 56, 57] ainsi que de van der Hoeven [234, 235, 236, 237]. Le plus important est peut-être la méthode d'évaluation numérique baptisée *bit burst* [57]. Elle appartient à la famille des algorithmes de sommation de séries D-finies par *scindage binaire*, que l'on peut faire remonter à une note de Schroeppel et Salamin [15, §178], et généralise par ailleurs des algorithmes proposés auparavant par Brent [37] pour des fonctions élémentaires particulières. Je renvoie à Bernstein [24, §12.7] et van der Hoeven [239, §1] pour plus de contexte et d'histoire.

Plusieurs algorithmes mieux connus (notamment grâce aux descriptions et implémentations de Haible et Papanikolaou [122] ainsi que de Gourdon et Sebah [115]) peuvent se voir comme des cas particuliers de l'algorithme *bit burst*. Contrairement à eux, l'algorithme général n'avait apparemment jamais été utilisé en pratique avant ce travail. Une explication possible est que dans la description de Chudnovsky et Chudnovsky, le contrôle des erreurs ne fait pas partie de l'algorithme. Il s'agit donc d'une méthode pouvant être spécialisée pour calculer n'importe quelle fonction D-finie donnée, moyennant des bornes d'erreur calculées séparément. La version de van der Hoeven [235] corrige cette faiblesse et fournit ainsi un véritable algorithme général d'évaluation à grande précision pour toute la classe des fonctions D-finies.

Toutes ces techniques s'étendent au calcul de limites et « conditions initiales généralisées » aux singularités de l'équation différentielle qui définit la fonction. Le cas des points singuliers réguliers est abordé dans plusieurs articles de Chudnovsky et Chudnovsky [53, 54, 57] puis traité de façon plus explicite par van der Hoeven [236], qui résout aussi par la suite celui (non implémenté dans NumGfun) des points singuliers irréguliers [239].

Sur le plan des implémentations, des bibliothèques généralistes comme CLN [121] ou MPFR [183] utilisent des routines à base de scindage binaire pour évaluer certaines fonctions élémentaires et fonctions spéciales. La sommation par scindage binaire de séries à convergence rapide est aussi l'algorithme de prédilection des logiciels dédiés au calcul à très grande précision de constantes remarquables sur des machines de bureau, dont les détenteurs récents du record de calcul de π [17, 262]. Enfin, la variété déjà impressionnante de fonctions spéciales disponibles nativement dans les systèmes de calcul formel n'est pas toujours suffisante dans les applications. Plusieurs articles récents sont consacrés à l'évaluation de classes de fonctions spéciales « un peu moins communes » qui recoupent celle couverte par NumGfun [4, 89].

Les fonctionnalités de calcul de *bornes symboliques* offertes par NumGfun sont uniques en leur genre. L'outil existant le plus proche est sans doute le logiciel ACETAF de Eble et Neher [82], dont l'idée est, grossièrement, de borner les coefficients de Taylor par une application en arithmétique d'intervalles de la formule intégrale de Cauchy. L'algorithme utilisé dans NumGfun emprunte à des techniques d'asymptotique automatique [93, 94] ainsi qu'à des algorithmes proposés par van der Hoeven pour l'analyse

effective [235, 237, 240].

Nous reviendrons plus en détail sur tout cela dans les chapitres correspondants du manuscrit (voir notamment §5.1.1, §6.1.2, §7.3 et chapitre 8).

Plan du chapitre. Le gros de ce chapitre est occupé par la section 1.2, qui illustre les possibilités de NumGfun a travers une session d'utilisation. Je présente ensuite brièvement l'architecture du module ainsi que quelques développements en cours ou en projet. Il me faut ici souligner que ce chapitre ne vise ni à servir de manuel utilisateur à NumGfun, ni à décrire en détail son fonctionnement. L'utilisation du module est décrite dans les pages d'aide Maple qui l'accompagnent (`?NumGfun` à l'invite de commande). La théorie sous-jacente, et quelques notes sur l'implémentation de certaines fonctionnalités, font l'objet d'une bonne partie de cette thèse. Le lecteur intéressé par les détails de l'implémentation pourra enfin se reporter au code source et à ses commentaires.

Conventions relatives aux exemples. Ce document contient un certain nombre d'exemples de code Maple, présentés de la façon suivante :

```
> evalf[25](exp(Pi*sqrt(163)));

   262537412640768744.0000000
```

Pour l'essentiel, ils sont exécutables en l'état. Les commandes

```
> with(gfun): with(NumGfun):
```

permettant d'accéder aux procédures et autres objets exportés par ces modules sans les qualifier par le nom du module sont sous-entendues dans tous les exemples. La sortie est souvent écourtée ou reformatée à la main pour des raisons de lisibilité.

Outre les fonctions publiques de Maple, gfun et NumGfun, nous ferons parfois appel à des fonctionnalités non documentées de NumGfun, et notamment au sous-module caché `NumGfun:-utilities`.

Les temps de calcul indiqués se réfèrent à des expériences menées sur un ordinateur équipé d'un processeur quadri-cœur AMD Athlon II X4 620 cadencé à 3 GHz et de 4 Go de mémoire vive, fonctionnant sous Debian `sid` avec un noyau Linux 3.0.0-rc2. Les calculs, ou du moins leur partie coûteuse, n'utilisent qu'un des cœurs. La version de Maple utilisée est Maple 13, qui fait appel pour les calculs en nombres entiers à GMP 4.2.1.

1.2 Une session avec NumGfun

1.2.1 gfun

Présentation. Le module gfun auquel se greffe NumGfun est une boîte à outils pour la manipulation de fonctions D-finies et de suites P-récursives. Pour citer sa documentation [110] :

> The gfun package has been designed as a help for the manipulation and discovery of functions or sequences satisfying linear differential or recurrence equations. The name of the package comes from its combinatorial application to generating functions.
>
> The basic principle of the package is that linear differential equations or recurrences can be used as data-structures to represent their solutions.

Les variables et procédures exportées par gfun sont les suivantes.

```
> with(gfun);

   [Laplace, NumGfun, Parameters, algebraicsubs, algeqtodiffeq, algeqtoseries, algfuntoalgeq, borel,
   cauchyproduct, diffeq*diffeq, diffeq+diffeq, diffeqtohomdiffeq, diffeqtorec, guesseqn, guessgf,
```

hadamardproduct, holexprtodiffeq, invborel, listtoalgeq, listtodiffeq, listtohypergeom, listtolist, listtoratpoly, listtorec, listtoseries, maxdegcoeff, maxdegeqn, mindegcoeff, mindegeqn, nth_term, pade2, poltodiffeq, poltorec, ratpolytocoeff, rec*rec, rec+rec, rectodiffeq, rectohomrec, rectoproc, seriestoalgeq, seriestodiffeq, seriestohypergeom, seriestolist, seriestoratpoly, seriestorec, seriestoseries]

Grossièrement, ses fonctionnalités s'organisent en :

- des opérations sur les séries représentées par des équations différentielles ou les suites représentées par des récurrences ;
- des fonctions pour passer d'une suite à sa série génératrice, et inversement ;
- des outils d'interpolation capables de « deviner » une suite à partir de quelques termes, qui renvoient une équation satisfaite par la suite ou sa série génératrice, ou dans certains cas une expression explicite de cette dernière ;
- et diverses fonctions utilitaires permettant, par exemple, de calculer des termes d'une suite récurrente, ou encore de passer facilement de l'expression explicite d'une fonction à une représentation par équation différentielle.

NumGfun fait largement appel aux fonctions des deux premières catégories.

Par exemple, l'exponentielle et la fonction d'Airy Ai sont des fonctions D-finies. Récupérons dans les tables de gfun des équations différentielles qu'elles satisfont :

```
> diffeq[exp] := prettify(holexprtodiffeq(exp(z), y(z)));
```

$$\left\{\frac{\mathrm{d}}{\mathrm{d}z}y(z)-y(z)=0,\ y(0)=1\right\}$$

```
> diffeq[Ai] := prettify(holexprtodiffeq(AiryAi(z), y(z)));
```

$$\left\{\frac{\mathrm{d}^2}{\mathrm{d}z^2}y(z)-z\,y(z)=0,\ y(0)=\frac{1}{3}\frac{\sqrt[3]{3}}{\Gamma\left(\frac{2}{3}\right)},\ (y')(0)=-\frac{1}{2}\frac{\sqrt[6]{3}\,\Gamma\left(\frac{2}{3}\right)}{\pi}\right\}$$

et utilisons gfun pour mener quelques calculs élémentaires avec ces équations. (La fonction `prettify` ne fait partie ni de Maple ni de gfun. C'est une commande *ad hoc* définie pour mettre certains résultats de ce mémoire sous une forme plus lisible.)

Opérations algébriques. Il est bien connu que la classe des fonctions D-finies est close par un certain nombre d'opérations, que l'on peut réaliser algorithmiquement sur les équations différentielles. Nous en ferons l'inventaire au chapitre 3. Par exemple, la commande `‘diffeq+diffeq‘` de gfun calcule une équation annulant la somme de deux fonctions D-finies données, ici $e^z+\mathrm{Ai}(z)$:

```
> prettify(‘diffeq+diffeq‘(diffeq[Ai], diffeq[exp], y(z)));
```

$$\left\{(z-1)\left(\frac{\mathrm{d}^3}{\mathrm{d}z^3}y(z)\right)-z\left(\frac{\mathrm{d}^2}{\mathrm{d}z^2}y(z)\right)-z\,(z-1)\left(\frac{\mathrm{d}}{\mathrm{d}z}y(z)\right)+(-z+1+z^2)\,y(z)=0,\right.$$
$$\left.y(0)=\frac{1}{3}\frac{\sqrt[3]{3}+3\,\Gamma\left(\frac{2}{3}\right)}{\Gamma\left(\frac{2}{3}\right)},\ (y')(0)=-\frac{1}{2}\frac{\sqrt[6]{3}\,\Gamma\left(\frac{2}{3}\right)-2\,\pi}{\pi},\ (\mathrm{D}\circ\mathrm{D})(y)(0)=1\right\}$$

De même, `algebraicsubs` calcule la composée d'une fonction D-finie et d'une fonction rationnelle ou algébrique. L'équation différentielle

```
> prettify(algebraicsubs(diffeq[Ai], 2*z*y=z^2+1, y(z)));
```

$$8\,(z-1)\,(z+1)\,z^5\left(\frac{\mathrm{d}^2}{\mathrm{d}z^2}y(z)\right)-16\,z^4\left(\frac{\mathrm{d}}{\mathrm{d}z}y(z)\right)-(z^2+1)\,(z-1)^3\,(z+1)^3\,y(z)$$

est ainsi satisfaite par $\mathrm{Ai}\left(\frac{1}{2}\,(z+z^{-1})\right)$.

Équations différentielles et récurrences. Seconde propriété de base : la suite des coefficients du développement en série d'une solution d'équation différentielle linéaire à coefficients polynomiaux, par exemple

```
> series(AiryAi(z), z, 7);
```

$$\frac{1}{3}\frac{\sqrt[3]{3}}{\Gamma\left(\frac{2}{3}\right)}-\frac{1}{2}\frac{\sqrt[6]{3}\,\Gamma\left(\frac{2}{3}\right)z}{\pi}+\frac{1}{18}\frac{\sqrt[3]{3}\,z^3}{\Gamma\left(\frac{2}{3}\right)}-\frac{1}{24}\frac{\sqrt[6]{3}\,\Gamma\left(\frac{2}{3}\right)z^4}{\pi}+\frac{1}{540}\frac{\sqrt[3]{3}\,z^6}{\Gamma\left(\frac{2}{3}\right)}+O\left(z^7\right)$$

satisfait une récurrence linéaire à coefficients polynomiaux aisément calculable à partir de l'équation différentielle. En l'occurrence,

```
> rec := prettify(diffeqtorec(diffeq[Ai], y(z), u(n)));
```

$$\left\{-u\left(n\right)+\left(n+3\right)\left(n+2\right)u\left(n+3\right)=0,u\left(0\right)=\frac{1}{3}\frac{\sqrt[3]{3}}{\Gamma\left(\frac{2}{3}\right)},u\left(1\right)=-\frac{1}{2}\frac{\sqrt[6]{3}\,\Gamma\left(\frac{2}{3}\right)}{\pi},u\left(2\right)=0\right\}$$

convient. La commande `rectodiffeq` effectue l'opération inverse de `diffeqtorec` — ou presque : voir §3.3 pour plus de détails.

Calcul d'un terme. Une fois obtenue cette récurrence, nous pouvons la « dérouler » pour obtenir rapidement un coefficient arbitraire. La fonction `rectoproc` de gfun fournit une *procédure Maple* qui prend en entrée un entier n et renvoie la valeur u_n.

```
> coeffproc := rectoproc(rec, u(n)):
> coeffproc(50001);
```

$$\frac{1}{8742085030\ldots(142143\text{ chiffres})\ldots0000000000}\frac{3^{1/3}}{\Gamma\left(\frac{2}{3}\right)}$$

Le calcul prend 1,9 secondes. Pour de « gros » résultats comme celui-ci, calculer de proche en proche $u_3, u_4, \ldots, u_n$ serait inefficace ; la procédure produite par `rectoproc` fait ici appel à NumGfun, qui implémente une autre méthode, à base de scindage binaire. L'algorithme — classique — est rappelé et étudié au chapitre 7.

1.2.2 NumGfun

NumGfun étend gfun par des outils liés à l'évaluation des suites et fonctions manipulées. Ses fonctionnalités essentielles sont

- d'une part, *l'évaluation* des fonctions D-finies à grande précision ;
- et d'autre part, le calcul de *bornes* sur les fonctions D-finies et les suites P-récursives.

Évaluation s'entend ici dans un sens assez large, qui va du calcul d'un terme d'une suite récurrente à la décomposition d'une solution d'équation différentielle sur une base de solutions définies au voisinage d'un point singulier, en passant par son « suivi » le long d'un chemin de prolongement analytique.

Les fonctionnalités se limitent en revanche à ce qu'on pourrait appeler le domaine des séries convergentes. L'étude des équations différentielles au voisinage de points singuliers irréguliers, les techniques d'évaluation à base de développements asymptotiques, la sommation au plus petit terme, la resommation de séries divergentes sont pour l'instant tous hors du champ des problèmes traités.

Voici la liste complète des fonctions publiques. Nous allons maintenant illustrer leurs possibilités sur quelques exemples.

```
> with(NumGfun);
```

[analytic_continuation, bound_diffeq, bound_diffeq_tail, bound_ratpoly, bound_rec, bound_rec_tail, diffeqtoproc, dominant_root, evaldiffeq, fnth_term, local_basis, needed_terms, transition_matrix]

1.2.3 Évaluation numérique

Une fonction spéciale pas si commune. La fonction de Heun doublement confluente[3] de paramètres a, b, c, d est la solution de l'équation différentielle

```
> diffeq := prettify(
      { diff(y(z),z,z)
          - (-2*z^5+4*z^3+z^4*a-2*z-a)/((z-1)*(z+1))^3 * diff(y(z),z)
          - (-z^2*b+(-c-2*a)*z-d)/((z-1)^3*(z+1)^3) * y(z),
      y(0)=1, D(y)(0)=0 });
```

$$\left\{ \frac{\mathrm{d}^2}{\mathrm{d}z^2} y(z) - \frac{(a\,z^2 - 2\,z^3 + a + 2\,z)\left(\frac{\mathrm{d}}{\mathrm{d}z} y(z)\right)}{(z+1)^2\,(z-1)^2} + \frac{(b\,z^2 + 2\,a\,z + c\,z + d)\,y(z)}{(z-1)^3\,(z+1)^3},\right.$$
$$\left. y(0) = 1, (y')(0) = 0 \right\}$$

Les fonctions de Heun comptent parmi les plus générales des fonctions spéciales que des logiciels comme Maple traitent nativement. Prenons des valeurs simples arbitraires pour les paramètres, et évaluons la fonction correspondante :

```
> a, b, c, d := 1, 1/3, 1/2, 3:
> evalf[51](HeunD(a, b, c, d, 1/3));
```

1.23715744756395253918007831405821000395447403052069

Le même calcul avec NumGfun part bien entendu de l'équation différentielle. La fonction `diffeqtoproc` est l'analogue pour les équations différentielles de la fonction `rectoproc` de gfun utilisée plus haut. Elle renvoie une procédure Maple qui prend en entrée un point z et une précision n, et calcule la valeur en z de la solution de l'équation différentielle, à précision 10^{-n}.

```
> myHeunD := diffeqtoproc(diffeq, y(z)):
> myHeunD(1/3, 50);
```

1.23715744756395253918007831405821000395447403052075

On observe que les derniers chiffres des résultats diffèrent d'un calcul sur l'autre. J'ai déjà mis l'accent mis sur le souci, dans NumGfun, de garantir la correction des résultats. Le lecteur ne sera donc pas surpris d'apprendre que la seconde valeur est la bonne, ce dont on se convainc en poussant le calcul à une plus grande précision.

```
> myHeunD(1/3, 160);
```

1.23715744756395253918007831405821000395447403052**747**2497736812233991047927263427910426036691704686822432669322058740005957868869065637255063771378117634825003548

Poursuivons avec un exemple un peu plus difficile. Le voisinage des singularités est souvent parmi les « endroits intéressants » où étudier une fonction, mais aussi parmi ceux où son évaluation est délicate. La fonction y solution de notre équation différentielle présente des singularités en $z = \pm 1$. Essayons de nous en approcher.

```
> evalf(HeunD(a, b, c, d, -0.9)), myHeunD(-0.9, 9);
```

2.695836**763**, 2.695836**219**

```
> evalf(HeunD(a, b, c, d, -0.99));
```

```
Warning, breaking after 2000 terms, the series is not converging
```

undefined

```
> myHeunD(-0.99, 400);
```

3. Dans une de ses deux définitions classiques, celle adoptée dans Maple [168].

```
4.677558527966890481646371616414130565650323560409922037183582493975621616831723241\
0744707789241015929982135365224156265638970467441803028111923987026650826169415 10\
9809652226279375975050987046539426225128475617116795496567630687966048899822188551\
1043494136629459587123627365393980067834480595323421947266813508293676138629023775\
8289885777340602080597240804541929600565356508117351708467455758748170258
```

Ce dernier calcul prend plusieurs secondes (5,2 s sur ma machine de test). On paie donc en temps de calcul la difficulté intrinsèque, sans compromis sur la correction du résultat. Des exemples comme celui-ci plaident en faveur de l'approche « générique » adoptée dans NumGfun, qui traite de larges classes de fonctions à égalité : moins de code spécifique signifie souvent moins de bugs !

Le fonctionnement de l'évaluateur numérique de NumGfun est détaillé au chapitre 8. Nous allons expérimenter plusieurs autres de ses fonctionnalités dans les quelques paragraphes à venir.

Un exemple « au hasard ». Au-delà des fonctions spéciales plus ou moins usuelles, on peut aussi bien sûr manipuler des fonctions D-finies arbitraires. Nous sortons là des fonctionnalités disponibles (à ma connaissance) dans d'autres logiciels. Tirons une équation différentielle au hasard.

```
> RandomTools:-SetState(state=42):
> random_diffeq := proc(ord, d) local myrat := 'rational'('denominator'=60); uses
  RandomTools; { add(Generate('polynom'(myrat, z, 'degree'=d)) * diff(y(z), [z$k]),
  k=0..ord), seq((D@@k)(y)(0) = Generate(myrat), k=0..ord-1) } end proc:
> rnd_diffeq := random_diffeq(4, 3);
```

$$\left\{\left(\frac{43}{60}-\frac{2}{15}z+\frac{11}{20}z^2-\frac{3}{4}z^3\right)y(z)+\left(\frac{47}{60}+\frac{1}{5}z+\frac{1}{60}z^2-\frac{13}{20}z^3\right)\left(\frac{\mathrm{d}}{\mathrm{d}z}y(z)\right)+\left(\frac{43}{60}+\frac{23}{60}z+\frac{9}{20}z^2+\frac{1}{4}z^3\right)\left(\frac{\mathrm{d}^2}{\mathrm{d}z^2}y(z)\right)+\left(\frac{1}{4}+\frac{7}{15}z+\frac{19}{20}z^2+\frac{2}{3}z^3\right)\left(\frac{\mathrm{d}^3}{\mathrm{d}z^3}y(z)\right)+\left(\frac{11}{15}-\frac{3}{5}z-\frac{19}{20}z^2-\frac{19}{30}z^3\right)\left(\frac{\mathrm{d}^4}{\mathrm{d}z^4}y(z)\right),\ y(0)=\frac{-7}{60},(y')(0)=\frac{-29}{30},(\mathrm{D}\circ\mathrm{D})(y)(0)=\frac{7}{15},\mathrm{D}^{\circ 3}(y)(0)=\frac{4}{5}\right\}$$

La fonction `evaldiffeq` effectue directement une évaluation en un point, sans passer par la création d'une procédure. Le point d'évaluation peut être réel ou complexe.

```
> evaldiffeq(rnd_diffeq, y(z), 1/2, 50);
```

$$-.52428724948743933011074780046842551144574795341755$$

```
> evaldiffeq(rnd_diffeq, y(z), (1+I)/3, 30);
```

$$-.449570759269227644270682723931-.260300150156116033712635106149\,\mathrm{i}$$

Comme indiqué en introduction, NumGfun emploie des algorithmes d'évaluation numérique de complexité essentiellement linéaire en le nombre de chiffres du résultat. Le but n'est pas de battre des records, et l'implémentation, qui privilégie presque toujours la généralité sur la rapidité, n'est pas compétitive — de plusieurs ordres de grandeur — avec les codes spécialisés qui reposent sur des méthodes apparentées. Se fonder sur un algorithme rapide est tout de même un gage de passage à l'échelle :

```
> evaldiffeq(rnd_diffeq, y(z), 1/3, 100000);
```

$$-.40784678373253882348505240495639163755195035646700552450754961526634594362120816\
4696978544698580070 1(\ldots)6862544805461487705749958969424450012994962704730390690372\
4271138075620527731404660378115130208441261227314328834453567063082866352718 7$$

Ce calcul prend environ 21 minutes. On pourrait vraisemblablement améliorer de façon considérable (quoique d'un facteur constant) les performances sur ce genre d'exemple en implémentant une variante plus soigneuse de l'algorithme de scindage binaire qui représente le gros du temps de calcul. À petite précision en revanche, le

temps passé dans la partie asymptotiquement coûteuse est négligeable devant celui, essentiellement indépendant de la précision, consacré au calcul de bornes d'erreur et à l'interprétation du code Maple. Ces questions de complexité et d'efficacité pratique sont étudiées au chapitre 7.

1.2.4 Prolongement analytique numérique

Matrices de transition. Que se passe-t-il si l'on spécifie une équation sans conditions initiales ? Considérons l'exemple suivant, qui reviendra à plusieurs reprises dans ce document.

```
> barediffeq := (z^2+1)*diff(y(z),z,z) + 2*z*diff(y(z),z);
```

$$(1+z^2)\left(\frac{\mathrm{d}^2}{\mathrm{d}z^2}y(z)\right)+2z\left(\frac{\mathrm{d}}{\mathrm{d}z}y(z)\right)$$

```
> evaldiffeq(barediffeq, y(z), (1+I)/2, 20);
```

$$1.0000000000000000000\,\_C_0+(.55357435889704525151+.40235947810852509365\,\mathrm{i})\,\_C_1$$

Ce qu'on obtient est une expression de la valeur de y au point $z_1=(1+i)/2$ en fonction des conditions initiales

$$(\_C_0,\_C_1)=\big(y(0),y'(0)\big).$$

On vérifie aisément que la fonction constante 1 et la fonction arctangente forment une base de solutions de l'équation considérée. Ces deux solutions de base correspondent respectivement aux conditions initiales $(1,0)$ et $(0,1)$. Le résultat de `evaldiffeq` est donc dans ce cas $1\cdot\_C_0+\arctan(z_1)\,\_C_1$. Chacun des coefficients de la combinaison linéaire est calculé à 10^{-20} près.

Nous pouvons pousser la même idée un peu plus loin et chercher la matrice qui fait passer d'un vecteur de conditions initiales à l'origine $\big(y(0),y'(0)\big)$ au vecteur de « conditions initiales en $z_1=1/2$ » $\big(y(z_1),y'(z_1)\big)$ correspondant :

```
> transition_matrix(barediffeq, y(z), 1/2, 30);
```

$$\begin{pmatrix} 1.0000000000000000000000000000000 & .46364760900080611621425623146 1 \\ 0. & .80000000000000000000000000000000 \end{pmatrix}$$

Cette matrice s'appelle *matrice de transition* ou *matrice résolvante* associée à l'équation $(1+z^2)\,y''+2\,z\,y'=0$ entre 0 et $\frac{1}{2}$. On reconnaît sur la deuxième ligne les valeurs de $1'=0$ et $\arctan' z=1/(1+z^2)$ en $z=\frac{1}{2}$.

Prolongement au-delà des points singuliers. Poursuivons le calcul avec les fonctions $z\mapsto 1$ et $z\mapsto\arctan z$, et essayons maintenant de les évaluer en $z=1$.

```
> evaldiffeq(barediffeq, y(z), 1, 30);
Error, (in gfun:-NumGfun:-ancont:-fail_if_singular_path) Step 0->1 may escape from
the disk of (guaranteed) convergence of the series expansions of the solutions of
-2*z*diff(y(z),z)+(-1-z^2)*diff(diff(y(z),z),z)
```

Échec : nous avons en effet passé sous silence une petite subtilité de la notion de « fonction » solution d'une équation différentielle employée dans NumGfun. La difficulté est liée à la présence de points singuliers de l'équation sur le cercle unité, en $\pm i$.

En effet, l'équation et les conditions initiales ne suffisent pas tout-à-fait, en général, à déterminer de façon unique une fonction analytique solution. Ainsi, la spécification

$$z\,y'(z)-1=0,\quad y(1)=0$$

pourrait aussi bien définir le logarithme complexe standard, analytique sur $\mathbb{C}\setminus\mathbb{R}_-$ et qui vérifie

$$\log(-1+i)=\frac{1}{2}\ln(2)+\frac{3}{4}\pi\,i,$$

que la fonction $z \mapsto \log(i\,z) - \frac{\pi}{2}\,i$, analytique sur $\mathbb{C} \setminus (i\mathbb{R}_-)$, qui prend la valeur

$$\frac{1}{2}\ln(2) - \frac{5}{4}\pi\,i$$

en $(-1+i)$. Le cas de l'arctangente n'est qu'une instance à peine plus compliquée du même phénomène, avec cette fois deux singularités, en $\pm i$, et deux demi-droites de coupure conventionnelles, qui partent des singularités vers l'infini le long de l'axe imaginaire.

Il est en revanche toujours possible de définir le *prolongement analytique* d'une solution le long d'un *chemin* évitant les points singuliers de l'équation. (Nous repoussons des rappels mathématiques un peu plus précis à ce sujet au chapitre 3.) La convention adoptée dans NumGfun est qu'un point d'évaluation seul fait référence à la solution définie sur le disque centré à l'origine et s'étendant jusqu'au point singulier le plus proche[4]. Dans les autres cas, il est nécessaire de donner à la fonction d'évaluation numérique un chemin de prolongement analytique. En remplaçant le point 1 par le chemin en ligne droite de 0 à 1, on obtient le résultat attendu :

```
> evalf[30](4 * evaldiffeq(barediffeq, y(z), [0,1], 30));
```

$$4.0000000000000000000000000000\ \_C_0 + 3.1415926535897932384626433832\ \_C_1$$

Un mot sur l'algorithme. Sur le disque où sont autorisées les évaluations sans chemin de prolongement explicite — et, sauf exception, seulement sur celui-ci —, le développement de Taylor à l'origine d'une solution de l'équation donnée en paramètre est toujours convergent. Il est donc possible, au moins en principe, de calculer la valeur de la fonction en sommant la série. (Ce n'est pas toujours une bonne idée en pratique !)

Cela ne fonctionne pas en l'état pour l'évaluation à l'extrémité d'un chemin quelconque. Mais connaissant les « conditions initiales » en un point intermédiaire z_1 qui caractérisent une certaine solution (initialement spécifiée par des conditions initiales à l'origine), on peut repartir de z_1 pour obtenir celles en un point z_2 un peu plus loin sur le chemin de prolongement, éventuellement en-dehors du disque de convergence du développement en série à l'origine. Et ainsi de suite : le prolongement analytique se résume au produit des matrices de transition le long de chaque tronçon.

En faisant afficher quelques informations sur le déroulement du calcul, nous voyons comment `evaldiffeq` subdivise automatiquement le chemin donné en plusieurs pas, le long desquels les matrices de transition sont données par des séries convergentes.

```
> infolevel[gfun] := 2:
  evaldiffeq(barediffeq, y(z), [0, 5/4*(1+I)], 20);
   `ordinary_step_transition_matrix:`, "87/100+87/100*I --> 5/4+5/4*I
   ord=1, prec~=.30240e-23, terms=136"
   `ordinary_step_transition_matrix:`, "0 --> 7/20+7/20*I
   ord=2, prec~=.60219e-23, terms=106"
   `ordinary_step_transition_matrix:`, "7/20+7/20*I --> 61/100+61/100*I
   ord=2, prec~=.28056e-23, terms=102"
   `ordinary_step_transition_matrix:`, "61/100+61/100*I --> 87/100+87/100*I
   ord=2, prec~=.60219e-23, terms=104"
   `absolute_precision_warning:`, "Recall that gfun:-NumGfun uses *absolute* error."
```

$$1.0000000000000000000\ \_C_0 + (1.13764519551855716794 + .351335639022646274521\,\mathrm{i})\ \_C_1$$

```
> infolevel[gfun] := 0:
```

On lit aussi la précision de calcul de chacune des matrices de transition intermédiaires, déterminée automatiquement à partir de l'erreur cible 10^{-20}, de même que le nombre de termes à retenir dans les développements en série. Les bornes sur les fonctions

4. Il serait malcommode de devoir systématiquement indiquer des lignes de coupure. Cependant, une version future de `diffeqtoproc` devrait accepter optionnellement la spécification de coupures délimitant un domaine de définition simplement connexe, et permettre alors l'évaluation directe en un point quelconque de ce domaine.

D-finies utilisées à cette fin sont établies au chapitre 6, sur la base des résultats du chapitre 5. L'algorithme de prolongement analytique et son implémentation, suivi des erreurs compris, font l'objet de la section 8.3.

1.2.5 Évaluation numérique, suite

Points de grande taille. La commande `evaldiffeq` et ses variantes sont limitées aux points à coordonnées rationnelles. Cela n'interdit pas de s'en servir pour évaluer les fonctions D-finies en des points plus généraux, mais on est conduit à remplacer le point d'évaluation par une approximation rationnelle à précision suffisante — rien d'inhabituel jusque-là.

Pour calculer $y(z)$ à 10^{-n} près avec n grand, on peut s'attendre à devoir connaître z lui-même à une précision du même ordre, par exemple à travers une approximation décimale avec un nombre de chiffres après la virgule linéaire en n. Or la complexité de l'algorithme utilisé pour sommer les séries, quasi-linéaire en la précision cible si le point d'évaluation est fixé, devient au moins de l'ordre de n^2 lorsque la taille de celui-ci et la précision du résultat sont tous deux d'ordre n. Cette faiblesse des techniques de scindage binaire est bien connue dans leurs applications classiques, aux fonctions hypergéométriques notamment.

NumGfun implémente un algorithme baptisé méthode *bit-burst*, qui rétablit la complexité quasi-linéaire. Le calcul suivant prend ainsi moins de 7 s, soit environ 25 fois moins que par pur scindage binaire.

```
> my_e := convert(evalf[5000](exp(1)), rational, exact);
```

$$\frac{2718281828459045235\ldots(4460\text{ chiffres})\ldots96873869117066666147}{1000000000000000000\ldots(4660\text{ zéros})\ldots00000000000000000000}$$

```
> diffeq := holexprtodiffeq(arctan(z), y(z)):
> evaldiffeq(diffeq, y(z), [0,my_e], 5000);
```

$$1.218282905017277621760\ldots(4660\text{ chiffres})\ldots85796212560201267299$$

L'algorithme *bit-burst* repose sur une utilisation astucieuse du prolongement analytique. Il est traité dans le chapitre consacré à ce dernier, en section 8.5.

Petite précision. Les fonctionnalités d'évaluation numérique de NumGfun sont centrées sur la grande précision. Bien souvent cependant, on n'a besoin d'évaluer une fonction donnée qu'à précision modérée, mais de façon répétée — par exemple pour tracer son graphe ou calculer numériquement une intégrale.

La commande `diffeqtoproc` offre un support limité des évaluations répétées. Elle accepte en paramètre optionnel une liste de disques sur lesquels précalculer des approximants polynomiaux de la fonction à évaluer, ensuite codés en dur dans la procédure d'évaluation. Celle-ci y fait appel lorsque le point d'évaluation et la précision donnés le permettent, ou se rabat sur l'algorithme général d'évaluation à grande précision à défaut. Le précalcul tire parti à la fois du moteur de calcul de bornes de NumGfun, pour garantir la précision des approximants, et du prolongement analytique, pour permettre de placer les disques n'importe où dans le plan complexe, y compris loin du point où sont données les conditions initiales.

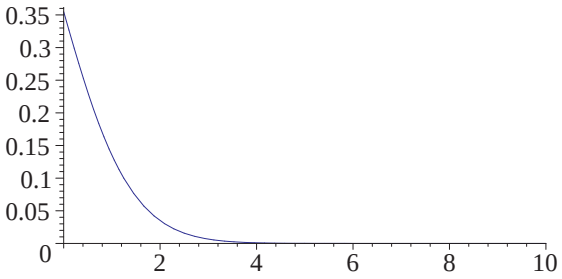
Dans l'exemple suivant, on trace le graphe sur $[0, 10]$ de la fonction d'Airy Ai grâce à un précalcul sur deux disques qui recouvrent cet intervalle.

```
> diffeq := holexprtodiffeq(AiryAi(z), y(z));
```

$$\left\{\frac{\mathrm{d}^2}{\mathrm{d}z^2}y(z) - z\,y(z),\, y(0) = \frac{1}{3}\frac{\sqrt[3]{3}}{\Gamma\left(\frac{2}{3}\right)},\, (y')(0) = -\frac{1}{2}\frac{\sqrt[6]{3}\,\Gamma\left(\frac{2}{3}\right)}{\pi}\right\}$$

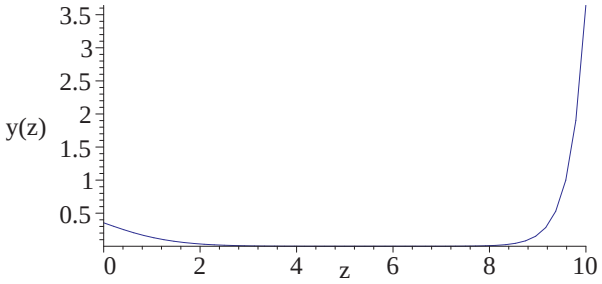
```
> myAi := diffeqtoproc(diffeq, y(z), prec=12, disks=[[2.5,2.6],[7.5,2.6]]):
```

```
> plot(myAi, 0..10, color=blue, thickness=3);
```



Comme un certain nombre de fonctions spéciales usuelles, la fonction Ai sur les réels positifs est une solution récessive de l'équation différentielle à travers laquelle on la manipule. Pour cette raison, une simple intégration numérique, comme celle effectuée par la fonction `DEplot` de Maple, donne des résultats grossièrement incorrects lorsque le temps d'intégration devient trop long.

```
> DEtools[DEplot](diffeq[1]=0, [y(z)], z=0..10, [[op(2..3,diffeq)]],
  'linecolor'=blue);
```



Nous reviendrons là-dessus en §8.7 de ce mémoire. Le chapitre 9 décrit quant à lui un algorithme pour calculer des approximants de meilleure qualité sur des segments, qui devrait être utilisé par `diffeqtoproc` à l'avenir.

1.2.6 Points singuliers réguliers

Les fonctions de Bessel modifiées I_ν et K_ν forment une base de solutions de l'équation différentielle

```
> diffeq := prettify(holexprtodiffeq(BesselI(nu,z), y(z)));
```

$$z^2\left(\frac{\mathrm{d}^2}{\mathrm{d}z^2}y(z)\right)+z\left(\frac{\mathrm{d}}{\mathrm{d}z}y(z)\right)-(\nu^2+z^2)\,y(z)$$

Celle-ci est singulière à l'origine. Bien que certaines des fonctions I_ν soient analytiques en zéro, on ne peut donc pas les définir par des conditions initiales en ce point de la même manière que les fonctions spéciales des exemples précédents. Il n'est pas non plus possible (du moins facilement) de trouver un autre point ordinaire, représentable exactement, en lequel la fonction soit donnée par des conditions initiales elles-mêmes représentables exactement.

Pourtant, I_ν et K_ν admettent des développements en série généralisée centrés en zéro et de rayon de convergence infini, par exemple [7, Eq. 9.6.12-13]

$$I_0(z)=\sum_{n=0}^{\infty}\frac{z^{2n}}{4^n\,n!^2},\qquad K_0(z)=\sum_{n=1}^{\infty}\left(\log\frac{z}{2}+H_n-\gamma\right)\frac{z^{2n}}{4^n\,n!^2}$$

où les H_n sont les nombres harmoniques. Un point singulier au voisinage duquel toute solution admet un développement convergent de ce genre s'appelle un point singulier régulier. Nous verrons au chapitre 4 comment définir et calculer une base canonique

de solutions au voisinage d'un tel point. Par exemple, avec $\nu = \sqrt{7}$, les premiers termes des développements des fonctions de la base canonique sont :

```
> local_basis(subs(nu=sqrt(7), diffeq), y(z), 0, 7);
```

$$\left[z^{(-\sqrt{7})} - \frac{1}{24}\left(\sqrt{7}+1\right) z^{(-\sqrt{7}+2)} + \frac{1}{192}\left(3+\sqrt{7}\right) z^{(-\sqrt{7}+4)} + \frac{1}{2304} z^{(-\sqrt{7}+6)} \left(8+3\sqrt{7}\right), z^{\sqrt{7}} + \frac{1}{24} z^{(\sqrt{7}+2)} \left(\sqrt{7}-1\right) - \frac{1}{192} z^{(\sqrt{7}+4)} \left(-3+\sqrt{7}\right) - \frac{1}{2304} z^{(\sqrt{7}+6)} \left(-8+3\sqrt{7}\right)\right]$$

Dès lors que l'on dispose d'une base de solutions convenable, la notion de matrice de transition s'étend. NumGfun est capable de résoudre numériquement des problèmes de connexion, c'est-à-dire de calculer les matrices de transition entre un point singulier régulier et un point ordinaire ou entre deux points singuliers réguliers. La matrice de transition de 0 à 1/3 regroupe ainsi les valeurs au point $z = 1/3$ des deux solutions de la base canonique en 0 affichées ci-dessus, ainsi que de leurs dérivées :

```
> NumGfun:-transition_matrix(diffeq, y(z), [0,1/3], 30);
```

$$\begin{pmatrix} 17.993511953679075830508666589332 & .550759619227735319578977572 98e-1 \\ -144.589027615738893901936888230190 & .439665592094415703409652498036 \end{pmatrix}$$

Une bonne manière de spécifier les fonctions de Bessel dans le cadre D-fini consiste en fait à donner leurs coordonnées dans cette base. Par exemple, $I_{\sqrt{7}}$ est un multiple simple de la seconde solution de base :

```
> cst := GAMMA(sqrt(7)+1)*exp(ln(2)*sqrt(7)):
> evalf[29](cst*BesselI(sqrt(7), 1/3)),
  evalf[30](cst*subs(z=1/3, diff(BesselI(sqrt(7), z), z)));
```

$$.550759619227735319578977572 98e-1, .439665592094415703409652498033$$

L'algorithme de connexion numérique, généralisation de celui de prolongement analytique déjà mentionné, est détaillé en §8.6. Son implémentation dans NumGfun est encore incomplète : essentiellement, seuls sont couverts les points singuliers à coordonnées rationnelles (autrement dit, appartenant à $\mathbb{Q}(i)$), et le contrôle de la précision dans l'implémentation est en partie heuristique.

1.2.7 Bornes symboliques

Nous avons entrevu plus haut (§1.2.4) que NumGfun calcule automatiquement des *bornes* sur les fonctions D-finies pour garantir la correction des évaluations numériques. Des bornes du même genre sont rendues disponibles à l'utilisateur sous forme d'expressions symboliques, à travers des fonctions dédiées. La forme des résultats et les critères de qualité à prendre en compte diffèrent légèrement, mais le moteur de calcul est partagé.

Suites P-récursives. En premier lieu, la commande `bound_rec` prend en entrée une récurrence qui définit une suite (u_n) à valeurs complexes, et renvoie l'expression explicite d'une suite « simple » (v_n) telle que $|u_n| \leqslant v_n$ pour tout n. La borne renvoyée est *fine*, au sens où v_n « ne s'éloigne pas trop » de $|u_n|$ quand $n \to \infty$. Limitons-nous à illustrer brièvement son utilisation :

```
> bound_rec({(2*n+2)^2*u(n+1) = -(n+1)*u(n), u(0) = 1}, u(n));
```

$$\frac{1000000001}{1000000000} \frac{(n+2)(n+1)\left(\frac{1}{4}\right)^n}{n!}$$

```
> bound_rec({4*u(n+1)-13*u(n+2)+5*u(n+3), u(0) = 17/5, u(1) = 3, u(2) = 3/12,
  u(3) = 0, u(4) = 5}, u(n));
```

$$\sum_{\alpha=\mathrm{RootOf}(4Z^2-13Z+5, 0.4457523585)} \frac{586725390387}{10000000000} \frac{\mathrm{RootOf}(4Z^2-13Z+5, 0.4457523585)\alpha^{-n}}{\alpha}$$

Asymptotiquement, ces deux bornes ne surestiment que d'un facteur polynomial en n les suites majorées. Les facteurs exponentiels ou en $n!$ sont optimaux. Nous verrons des exemples plus développés au début du chapitre 5, consacré à l'algorithme qui conduit à ces résultats.

Majorants pour les séries D-finies. Une variante de ce même algorithme fournit une *série majorante* d'une série D-finie quelconque, c'est-à-dire une série qui la domine coefficient par coefficient. La relation de domination permet de transférer certaines propriétés analytiques — à commencer par la convergence — du majorant à la série majorée. Les majorants que nous calculerons préservent notamment (sous une hypothèse de généricité) le rayon de convergence de la série qu'ils majorent, tout en étant suffisamment simples pour que les propriétés à transférer soient faciles à établir, à la main ou automatiquement. On a par exemple :

```
> bound_diffeq(holexprtodiffeq(arctan(z), y(z)), y(z));
```

$$\frac{1000000001}{2000000000}\,\frac{1}{(1-z)^2}$$

```
> bound_diffeq(holexprtodiffeq(erf(z), y(z)), y(z));
```

$$\frac{416666669}{5000000000}\sum_{n=0}^{\infty}\frac{(n^3+6\,n^2+11\,n+6)\,z^n}{\Gamma\left(\frac{1}{2}\,n+1\right)}$$

```
> bound_diffeq(y(z) = (z^7+3*z^2+z+1)/((z-2)^3*(z^3-3)*(z-I)^2), y(z));
```

$$\frac{352329590023}{5000000000000}\,\frac{1}{(-z+1)^2}+\frac{1262343}{40000000}\,z^2+\frac{102975491}{2000000000}\,z^3$$

La série arctangente, de rayon de convergence égal à 1, est majorée par une fraction rationnelle avec un pôle en 1. La fonction d'erreur, par une fonction entière qui met en évidence la décroissance « en $1/\sqrt{n!}$ » de ses coefficients. Une fraction rationnelle (solution d'une équation différentielle d'ordre zéro) est majorée par une fraction rationnelle avec un seul pôle, choisi d'ordre minimal.

Restes de séries. Parmi les plus utiles des résultats que l'on peut tirer d'une série majorante se trouvent des bornes sur les restes de la série majorée. Ainsi, la commande `bound_diffeq_tail` de NumGfun établit une borne supérieure sur l'erreur commise en tronquant une série y, en fonction du module $|z|$ du point d'évaluation et de l'ordre de troncature n. Concernant les deux fonctions spéciales simples dont nous venons de calculer des séries majorantes, on trouve :

```
> bound_diffeq_tail(holexprtodiffeq(arctan(z), y(z)), y(z), n);
```

$$-\frac{1000000001}{2000000000}\,\frac{(n\,|z|-n-1)\,|z|^n}{(|z|-1)^2}$$

```
> bound_diffeq_tail(holexprtodiffeq(erf(z), y(z)), y(z), n);
```

$$\begin{cases}\dfrac{1250000007}{1280000000000}\,\dfrac{\sqrt{2}\,(n+9)^4\left(\frac{\sqrt{2}\,|z|}{1-\frac{4}{n+9}}\right)^n\left(1+\frac{\sqrt{2}\,|z|}{1-\frac{4}{n+9}}\right)}{\Gamma\left(\frac{1}{2}\,n+1\right)\left(1-2\,\frac{|z|^2}{\left(1-\frac{4}{n+9}\right)^2(n+2)}\right)} & 4\,\dfrac{|z|^2}{\left(1-\frac{4}{2\,|z|^2+9}\right)^2}<n \\ \dfrac{3500000000000}{282842712051}\,\dfrac{8\,|z|^6-1}{\sqrt{2}\,|z|-1}+\dfrac{81}{4}\,\dfrac{|z|^6\,(2\,|z|^2-1)\,\mathrm{e}^{(|z|^2)}}{\sqrt{2}\,|z|-1} & \text{otherwise}\end{cases}$$

À défaut d'être parfaitement lisibles (!), ces bornes sont explicites (les formules ne font intervenir que des opérations usuelles) et satisfont elles aussi certaines propriétés de finesse. Cette fonctionnalité est fondée sur les résultats du chapitre 6. Ce sont essentiellement des bornes de ce type, avec un compromis finesse-simplicité un peu

différent (§6.3.2), qui sont au cœur du contrôle d'erreur dans le prolongement analytique numérique.

1.3 Architecture

Il ne vaut pas la peine de s'étendre longuement sur les détails de l'implémentation, qui n'est malheureusement pas, du moins en l'état, un modèle du genre ! Cette section donne cependant des informations générales sur le code ainsi que quelques repères pour le lecteur qui voudrait s'y orienter.

NumGfun réside dans le répertoire `NumGfun/` de la distribution source de gfun. Le module `NumGfun` est défini dans le fichier `main.mm`. Il se compose actuellement des sous-modules suivants, chacun défini dans un fichier `.mm` à son nom.

ancont. Évaluation numérique et prolongement analytique : scindage binaire *ad hoc* optimisé (§8.4), prolongement analytique (§8.3.2), réécriture des chemins dont *bit burst* (§8.3.2, §8.4.1).

bound_normal_diffeq. Calcul des paramètres de borne pour les séries D-finies normalisées (§5.3.3).

bound_ratpoly. Bornes sur les fractions rationnelles (§5.3.2).

bounds. Fonctions partagées de calcul de bornes : réduction au cas normalisé (§5.2.3, §5.4.1), formules explicites (§5.4.2, §6.2).

diffeqtoproc. Génération de procédures d'évaluation numérique (§8.7).

dominant_root. Racines dominantes de polynômes (non décrit dans ce mémoire, mais voir §5.2.1).

matrices. Fonctions utilitaires pour manipuler les matrices d'éléments de $\mathbb{Q}[i]$ avec numérateurs et dénominateurs séparés, et restes d'expérimentations autour du scindage binaire. Doit disparaître ou être entièrement réécrit.

nthterm. Calcul d'un terme d'une suite récurrente par scindage binaire (§7.3), fonctions auxiliaires pour `gfun:-rectoproc`.

numeric_bounds. Bornes pour l'évaluation numérique : ordres de troncature, bornes sur les matrices de transition (§6.3, §8.3.2, §8.4).

recasympt. Asymptotique de suites récurrentes (§8.7 ; en développement).

regsing. Connexion entre points ordinaires et points singuliers réguliers : base locale (§4.4.3), scindage binaire *ad hoc* et matrices de transition (§8.6).

Settings. Paramètres internes réglables par l'utilisateur.

symbolic_bounds. Bornes symboliques (§5.4.2, §6.2).

types. Vérifications de type.

utilities. Fonctions utilitaires (flottants, intervalles, nombres algébriques, opérateurs différentiels et de récurrence...).

La figure 1.1 résume les dépendances entre ces modules, et indique les fonctions publiques définies dans chacun. La plupart d'entre eux dépend par ailleurs de gfun. NumGfun a accès aux fonctions internes de gfun mais utilise très majoritairement son interface publique, ce qui le rend relativement facile à distribuer séparément ou à porter vers une autre bibliothèque similaire au besoin. Inversement, deux fonctions de gfun (`nth_term` et `rectoproc`) font appel à NumGfun.

Le code est relativement compact, avec un peu plus de 3000 lignes de Maple — un peu plus de 5000 en comptant les lignes blanches et les commentaires.

Il faut ajouter à cela une bibliothèque C en développement, `libNumGfun`, qui vise à servir de noyau compilé à NumGfun. Son utilisation est facultative. Quand elle est disponible, un certain nombre de fonctions internes de NumGfun peuvent être remplacées par des implémentations alternatives d'interface identique, qui appellent `libNumGfun` à travers le mécanisme d'appel de code externe de Maple. Cette bibliothèque n'en est qu'à ses débuts. Elle ne fait pas partie pour l'instant de la version distribuée publiquement de NumGfun, et n'est pas utilisée dans les exemples de ce mémoire, sauf mention contraire explicite.

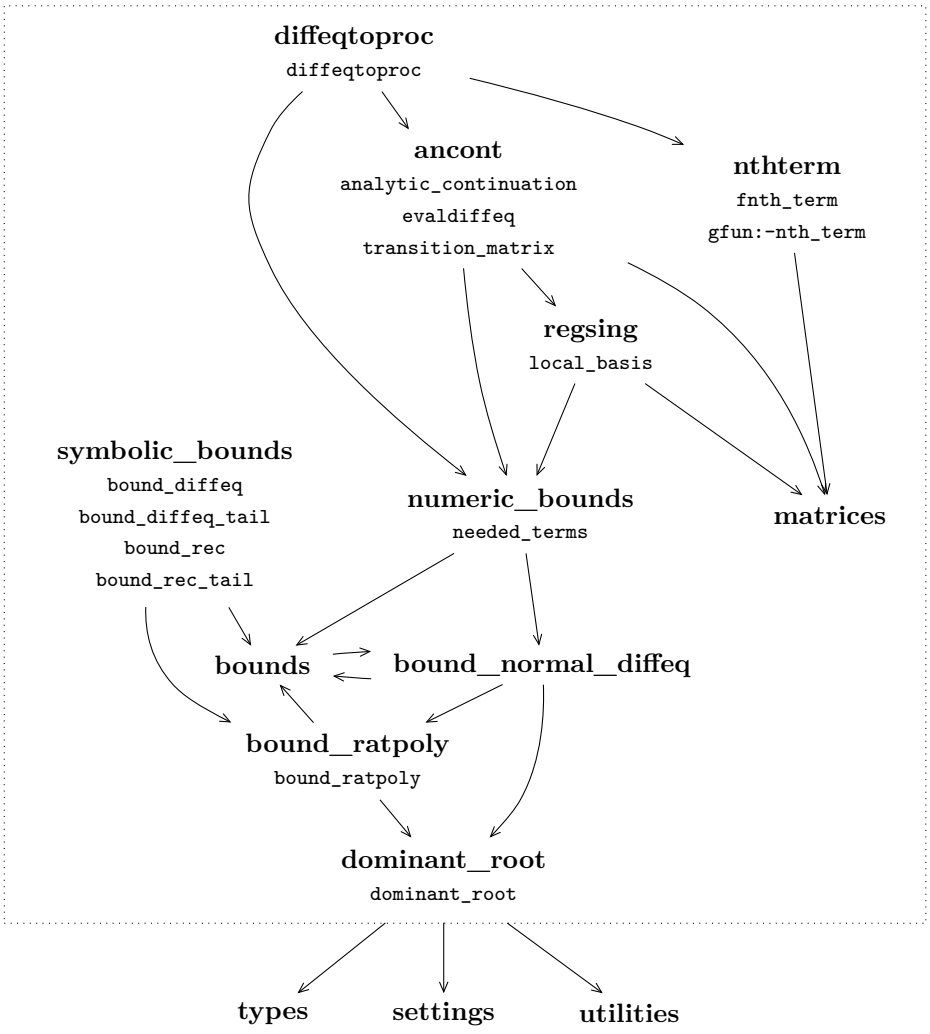


Figure 1.1. Les sous-modules de NumGfun et leurs dépendances.

1.4 Développements à venir

Je n'ai pas l'intention d'abandonner le développement de NumGfun avec la fin de ma thèse. Voici quelques chantiers déjà en cours.

1. Comme indiqué plus haut, l'implémentation de la connexion aux points singuliers réguliers est encore incomplète au regard des objectifs. Ses performances pourront sans doute être grandement améliorées une fois le code stabilisé.
2. Sur celle-ci s'appuie un début de module d'asymptotique automatique à base d'analyse de singularité. Ce n'est pour l'instant qu'un jouet, alors qu'il s'agit d'une des fonctionnalités qui suscitent le plus d'intérêt de la part des utilisateurs potentiels.
3. Le support de la petite précision est trop limité. En particulier, les fonctionnalités de précalcul de `diffeqtoproc` doivent devenir plus commodes d'utilisation, fournir des approximants plus compacts à qualité égale, et être étendues à d'autres domaines que des disques sur la base des idées du chapitre 9.
4. Point lié au précédent, je voudrais mener aussi rapidement que possible le noyau en C à un stade où il puisse être intégré (optionnellement) à la version standard de NumGfun. Le but est à la fois d'améliorer les performances du prolongement analytique numérique à précision modérée et de pouvoir expérimenter des variantes algorithmiques plus fines, pour lesquelles un langage de programmation interprété n'est pas approprié.

5. Enfin, l'ensemble de NumGfun a subi bien des adaptations au fur et à mesure qu'évoluait ma compréhension des algorithmes mais aussi des subtilités de Maple. Je ne suis pas toujours parvenu à moderniser ou élaguer au fur et à mesure les parties qui en avaient besoin, ni à contenir entièrement la duplication de code. Il y a une certaine incohérence dans la façon de manipuler les flottants, les intervalles ou encore les nombres algébriques, et dans les hypothèses que je fais sur la rigueur des fonctionnalités natives de Maple. Le stade actuel de développement semble adapté pour systématiser les approches employées avec une meilleure compréhension des forces et faiblesses de Maple.

L'objectif à plus long terme du développement de `libNumGfun` est en fait de réécrire toute une partie de NumGfun, en procédant « de bas en haut ». Cela devrait permettre de rendre certaines fonctionnalités accessibles sans dépendance à Maple, et notamment depuis des systèmes de calcul mathématique libres comme Sage [225] et Mathemagix [243]. La partie compilée est actuellement développée en C sur la base de la bibliothèque d'arithmétique multiprécision GMP. Mon projet est de l'étendre en faisant appel à des bibliothèques de calcul formel performantes comme FLINT [124] ou certains des modules de Mathemagix.

Chapitre 2

Le DDMF

« If Sesame Street can run commercials
for letters of the alphabet, I demand equal time
for my favorite special functions. »

— R. William Gosper [112]

Ce court chapitre présente une seconde réalisation logicielle à laquelle j'ai collaboré : le *Dynamic Dictionary of Mathematical Functions*, ou DDMF, un site web interactif sur les fonctions spéciales produit de façon complètement automatique à partir d'algorithmes de calcul formel et de modèles de document. Ses besoins ont motivé et orienté les travaux de cette thèse, et notamment le développement de NumGfun. Le DDMF est un travail en commun avec Alexandre Benoit, Frédéric Chyzak, Alexis Darrasse, Stefan Gerhold et Bruno Salvy, dans le cadre d'un projet du centre de recherche joint INRIA-Microsoft Research d'Orsay. Une partie du texte de ce chapitre est adapté de sa présentation à la conférence ICMS 2010 [19].

2.1 Motivation

Les formulaires de référence sur les fonctions usuelles, comme le célèbre *Handbook of Mathematical Functions* d'Abramowitz et Stegun [7], les trois volumes du « Bateman project » [87] ou encore la table d'intégrales et séries en six volumes de Prudnikov, Brychkov et Marichev [206] ont été un outil de travail quotidien pour des milliers de scientifiques et d'ingénieurs. Ces ouvrages encyclopédiques couvrent une large gamme de fonctions élémentaires et de fonctions spéciales, et répertorient toutes sortes de formules les concernant : équations différentielles, primitives et intégrales définies, inégalités, relations de récurrence, séries, développements asymptotiques, approximations, dans certains cas aussi graphes et tables numériques. Préparés par des spécialistes, ils ont été relus et vérifiés avec soin. Des dizaines de milliers de citations dans la littérature scientifique témoignent de leur succès.

Les premières éditions de ces livres datent d'il y a trente à soixante ans. Contrairement aux tables purement numériques, ils restent largement d'actualité. Mais l'arrivée du Web a récemment bouleversé la manière dont on accède à l'information. Conscient de cette évolution, le NIST a accompagné le *NIST Handbook of Mathematical Functions* [195], révision parue en 2010 du *Handbook* de 1964, d'un site web appelé *NIST Digital Library of Mathematical Functions* [3]. Celui-ci permet de naviguer parmi les formules, de les exporter dans différents formats, et dispose d'un moteur de recherche.

Dans le même temps, les systèmes de calcul formel sont passés du stade d'expérimentations balbutiantes à celui de bibliothèques robustes et conséquentes d'algorithmes mathématiques. Le plus souvent, l'implémentation des fonctions spéciales dans ces systèmes s'appuie largement sur une base de formules saisies depuis des ouvrages de référence comme ceux mentionnés plus haut. Pourtant, les algorithmes de calcul formel en sont aujourd'hui à un stade où une grande partie de ces formules pourraient être obtenues automatiquement.

Le projet DDMF vise à combiner ces algorithmes et l'interactivité permise par le

Web pour produire un « dictionnaire de fonctions spéciales » facile d'accès, généré automatiquement. Le dictionnaire est interactif au sens où il permet d'adapter formules et graphiques au besoin de l'utilisateur, de pousser à la demande les calculs de développements asymptotiques ou de valeurs numériques à des précisions arbitraires, ou encore d'afficher des preuves qui retracent de façon détaillée les calculs ayant conduit à tel ou tel résultat. Le DDMF fait suite à un premier prototype, statique et centré sur les développements asymptotiques, développé par Ludovic Meunier et Bruno Salvy [172].

La figure 2.1 reproduit une entrée complète du DDMF. Le site web lui-même est accessible à l'adresse `http://ddmf.msr-inria.inria.fr/`.

2.2 Algorithmes de calcul formel

Du point de vue du calcul formel, quelle définition de « fonction spéciale » se donner pour que toutes les informations souhaitées puissent être obtenues de façon algorithmique ? Notre choix dans le DDMF, déjà longuement présenté dans le début de cette thèse, est de nous concentrer sur les fonctions *données comme solutions d'équations différentielles ou de récurrences linéaires à coefficients polynomiaux*. Notre structure de données de base consiste essentiellement en une telle équation et des conditions initiales convenables. Dans l'exemple présenté en figure 2.1, il s'agit du contenu de la section 1 de la page.

Cette manière de représenter les fonctions spéciales a connu un certain succès en calcul formel, sur la base des travaux de Stanley [222], Lipshitz [159], Zeilberger [264], et plus récemment Chyzak, Salvy *et al.* [59, 60, 64]. Notamment, une grande partie des calculs sous-jacents aux pages du DDMF repose sur gfun [214].

À partir de cette structure de données, nous avons utilisé ou développé des algorithmes qui calculent toutes sortes de propriétés des fonctions. Ainsi, la section consacrée au développement à l'origine (section 4 de la page reproduite) s'appuie sur une récurrence satisfaite par les coefficients de Taylor, calculée à partir de l'équation différentielle (voir les rappels à ce sujet en §3.2-3.3). Lorsque les coefficients admettent une forme close comme terme hypergéométrique, ce qui est le cas dans notre exemple, celle-ci est déterminée en résolvant la récurrence par l'algorithme de Petkovšek [201]. Quoi qu'il en soit, la suite de la section donne les premiers termes du développement. Un formulaire permet de calculer l'ordre auquel le tronquer afin d'approcher la fonction à une précision et sur un disque donnés, sur la base des algorithmes développés dans les chapitres 5 et 6 de ce mémoire.

Plus bas, des calculs analogues sont mis en œuvre au voisinage de l'infini et, le cas échéant, des points singuliers de l'équation différentielle. Une section est consacrée aux expressions de la fonction en termes de fonctions hypergéométriques classiques et généralisées, une autre à son développement de Tchebycheff sur le segment $[-1;1]$. Les algorithmes derrière ces deux sections sont décrits dans la thèse d'Alexandre Benoit [18]. La dernière section donne un exemple de transformée intégrale.

En remontant au début de l'exemple, la section « *Numerical Evaluation* » joue le rôle des tables numériques dans les formulaires papier classiques. Un champ à remplir y donne accès à des approximations numériques garanties, à précision arbitraire, de valeurs de la fonction ou, le cas échéant, de son prolongement analytique. L'algorithme est celui du chapitre 8 de cette thèse, et l'implémentation celle de NumGfun. Quant au graphe affiché en tête de page, il devrait dans un avenir proche être tracé à partir des approximants polynomiaux étudiés au chapitre 9.

Home Glossary

The Special Function $\mathrm{Ai}(x)$

[-] 1. Differential equation

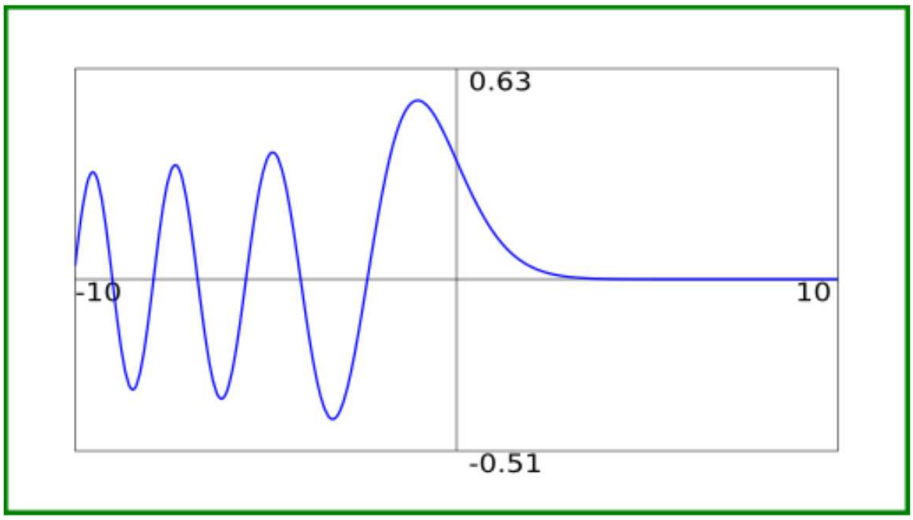
rendering link

The function $\mathrm{Ai}(x)$ satisfies

$$\frac{d^2}{dx^2}y(x) - xy(x) = 0$$

with initial values $y(0) = 1/3\,\frac{\sqrt[3]{3}}{\Gamma(2/3)}$, $\left(y'\right)(0) = -1/2\,\frac{\sqrt[6]{3}\Gamma(2/3)}{\pi}$.

[-] 2. Plot



min = -10 max = 10 Envoyer

[-] 3. Numerical Evaluation

$$\mathrm{Ai}(1/4 + 1/4i) \approx 0.28881085 - 0.06285935i.$$

(Below, `path` may be either a point z or a broken-line path $[z_1, z_2, \ldots, z_n]$ along which to perform analytic continuation of the solution of the defining differential equation. Each z_k should be of the form `x + y*i`.)

path = 1/4+1/4*i precision = 8 Envoyer

[-] 4. Taylor Expansion at 0

- Expansion:

$$\mathrm{Ai}(x) = \sum_{n=0}^{\infty} 1/3\,\frac{\sqrt[3]{3}x^{3n}}{9^n\,\Gamma(n+2/3)n!} + \sum_{n=0}^{\infty} -1/9\,\frac{3^{2/3}x^{3n+1}}{9^n\,\Gamma(n+4/3)n!}.$$

- See the recurrence relation for the coefficients of the Taylor expansion.

- First terms:

$$\mathrm{Ai}(x) = \left(1/3\,\frac{\sqrt[3]{3}}{\Gamma(2/3)} - 1/2\,\frac{\sqrt[6]{3}\,\Gamma(2/3)}{\pi}\,x + 1/18\,\frac{\sqrt[3]{3}}{\Gamma(2/3)}\,x^3\right) + O\left(x^4\right).$$

order = 4 Envoyer

- One gets a polynomial that approximates $\mathrm{Ai}(x)$ uniformly on the disk $|x| \le r = 3/10$ with absolute error less than 10^{-8} by retaining the terms up to x^{19} of this power series expansion. See this polynomial approximation. Proof based on the general tail bound.

r = 0.3 prec = 8 Envoyer

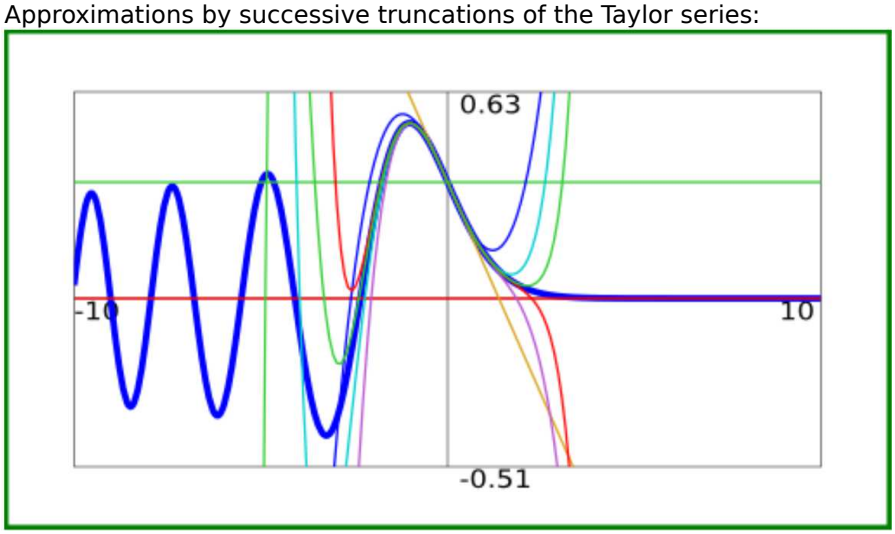
- A majorant series for $\mathrm{Ai}(x)$ in 0 is given by

$$\frac{177514027337}{500000000000}\sum_{n=0}^{\infty}\frac{3^{-2/3n}u_n x^n}{(\Gamma(1/3n+1))^2}$$

where u_n denotes the coefficient of z^n in the series expansion at the origin of

$$-e^{\frac{1}{10800000000000}x^3\left(6423529440000+1261764738600x^3+407843172800x^6+143382389925x^9+36705920208\right.}$$

- Approximations by successive truncations of the Taylor series:



min = -10 max = 10 Envoyer

[-] 5. Local Expansions at Singularities and at Infinity

The differential equation above has 0 non-zero finite singular points.

- At ∞:
  - Expansion:

$$\mathrm{Ai}(x) = 2\,e^{-2/3\left(x^{-1}\right)^{-3/2}}\left(x^{-1}\right)^{7/4}\sum_{n=0}^{\infty} -\frac{1}{96}\frac{(12n+5)(12n+1)!\left(x^{-1}\right)^{3n}}{331776^n\sqrt{\pi}(6n)!(4n)!(2n+1)} + e^{-2/}$$

- See the recurrence relation for the local coefficients.
- First terms:

$$\mathrm{Ai}(x) = e^{-2/3\left(x^{-1}\right)^{-3/2}}\sqrt[4]{x^{-1}}\left(1/2\,\frac{1}{\sqrt{\pi}} - \frac{5}{96}\frac{\left(x^{-1}\right)^{3/2}}{\sqrt{\pi}} + \frac{385}{9216}\frac{1}{\sqrt{\pi}x^3} + O\left(\left(x^{-1}\right)\right.\right.$$

order = 4 Envoyer

[-] 6. Hypergeometric Representation

$$\mathrm{Ai}(x) = -1/2\,\frac{x^6\sqrt{3}\Gamma\left(2/3\right){}_0F_1\left(;\ 4/3;\ 1/9x^3\right)}{\pi} + 1/3\,\frac{\sqrt[3]{3}\,{}_0F_1\left(;\ 2/3;\ 1/9x^3\right)}{\Gamma(2/3)}.$$

$$\mathrm{Ai}(x) = e^{-2/3\left(x^{-1}\right)^{-3/2}}\sqrt[4]{x^{-1}}\,{}_4F_1\left(1/12,\ \frac{5}{12},\ \frac{7}{12},\ \frac{11}{12};\ 1/2;\ 9/4x^{-3}\right)\frac{1}{\sqrt{\pi}} - \frac{5}{96}e^{-2/3\left(x^{-1}\right)^{-3/2}}$$

[-] 7. Chebyshev Expansion over $[-1, 1]$

- Chebyshev expansion:

$$\mathrm{Ai}(x) = \frac{1}{192}\left(-3\sqrt[6]{3}(\Gamma(2/3))^2\,{}_1F_4\left(5/6;\ 1,\ 4/3,\ 5/3,\ 5/3;\ \frac{1}{1296}\right) + 64\sqrt[3]{3}\,{}_1F_4\left(1/6;\ 1\right.\right.$$

- First terms and polynomial approximation:

$$\mathrm{Ai}(x) = 0.347553T_0(x) - 0.214710T_1(x) - 0.00986197T_2(x) + 0.0146332T_3(x) - 0.002327$$

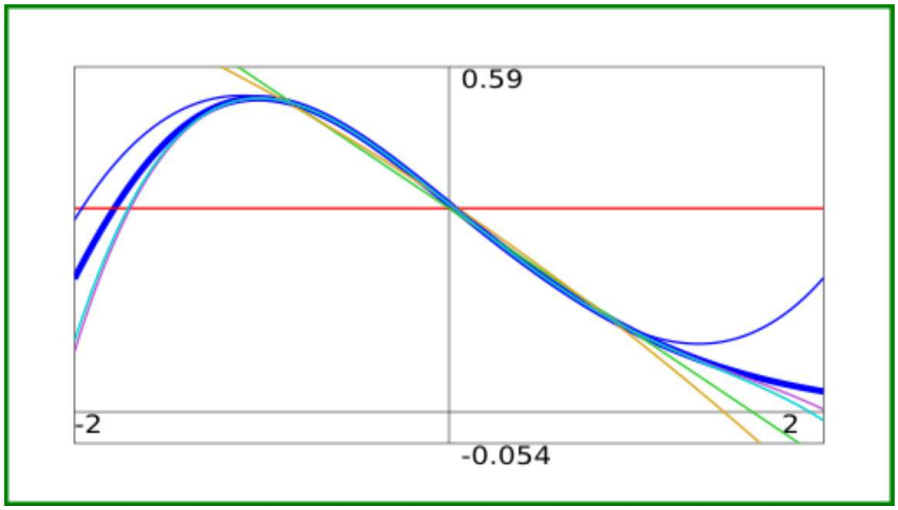
$$\mathrm{Ai}(x) \approx 0.3550262926 - 0.2588712070x - 0.0000029468x^2 + 0.0595792280x^3 - 0.02155$$

order = 6 Envoyer

- The coefficients c_n in the Chebyshev expansion $\mathrm{Ai}(x) = \sum_{n=0}^{\infty} c_n T_n(x)$ satisfy the recurrence

$$(-n-4)c(n) + (n+2)c(n+2) + \left(8n^3 + 72n^2 + 208n + 192\right)c(n+3) + (n+4)c(n+4) + (\cdot$$

- Approximations by successive truncations of the Chebyshev series on $[-2, 2]$:



- Errors of approximation by successive truncations of the Chebyshev series on $[-1, 1]$:

[-] 8. Laplace Transform

For $\Re(s) > 0$, the Laplace transform satisfies the following linear differential equation:

$$6s^2\,\Gamma(2/3)\pi\frac{d}{ds}y(s) + 6s\,\Gamma(2/3)\pi\left(s^3-2\right)y(s) - s^4\left(2\sqrt[3]{3}\pi s - 3(\Gamma(2/3))^2\sqrt[6]{3}\right) = 0.$$

Figure 2.1. Exemple de description d'une fonction spéciale dans le DDMF, version 1.6 (novembre 2010).

2.3 Présentation en ligne et interactivité

Les pages qui constituent le DDMF sont programmées au moyen d'un outil spécifique appelé DynaMoW (Dynamic Mathematics on the Web) [61], développé par Frédéric Chyzak et Alexis Darrasse. DynaMoW prend la forme d'une bibliothèque accompagnée d'extensions syntaxiques pour le langage Objective Caml.

Côté calcul formel, DynaMoW permet un interfaçage transparent avec différents systèmes symboliques, principalement Maple en l'état actuel des choses. Les requêtes à ces systèmes renvoient des fragments de formules mathématiques repris dans le document engendré, mais aussi des résultats manipulables par le programme DynaMoW. Côté document, DynaMoW offre des outils pour assembler en des pages web structurées les résultats des calculs et les portions de texte ou de formules embarqués dans le code source.

```
let service TaylorTruncation
  (sf_id : string)
  (deq : maple)
  (r : string = "0.3")
  (prec : int = 8) :
  DC.sec_entities * (int option) with { title = title } =
  let rep = SF.rep_of_id sf_id in
  try
    let rad = symbolic_r r in
    let infsing =
      << gfun:-NumGfun:-utilities:-diffeq_infsing($(deq), y(x)) >>
    in
      if <:bool< signum($(rad) - abs($(infsing))) >= 0 >> then
        <:par< It is not easy to determine whether <:isymb< $(rep) >>
        can be evaluated on the disk <:imath< |x| \leq $(symb:rad) >> using
        this power series expansion, since the said disk contains
        <:imath< $(symb:infsing) >>, which is a singular point
        of the differential equation above. >>,
        None
      else
        let trunc_order = <:int< gfun:-NumGfun:-needed_terms($(deq), y(x),
                                    0, 0, $(rad), 10^(-$(int:prec))) >> in
        let pa =
          DC.link_service
            (PolynomialApproximation.url (sf_id, trunc_order) None)
            <:text<polynomial approximation>>
        and tb =
          DC.link_service
            (GeneralFormulaForTaylorBound.url (sf_id, deq) None)
            <:text<tail bound>>
        in
          <:par< One gets a polynomial that approximates
            <:isymb< $(rep) >>
          uniformly on the disk
            <:imath< \mathopen|x\mathclose| \leq r = $(symb:rad) >>
          with absolute error less than
            <:imath< 10^{ $(int:-prec) } >>
          by retaining the terms up to
            <:imath< x^{$(int:(trunc_order - 1))} >>
          of this power series expansion.  See this $(pa).
          Proof based on the general $(tb). >>,
          (Some trunc_order)
  with (Warning msg) -> (DC.warning msg, None)
```

Figure 2.2. Extrait du code source du DDMF.

Les documents construits sont des objets du langage. Cela signifie que la structure du document elle-même peut dépendre des résultats de calculs exécutés par les systèmes symboliques. Le nombre et le type des items d'une entrée du DDMF consacrés à l'asymptotique d'une fonction sont par exemple déterminés par le calcul des singularités de l'équation différentielle. Ainsi, les sources DynaMoW ne sont pas de simples textes à trous statiques, mais, au besoin, des programmes à part entière.

Une telle architecture facilite la production de « preuves » (traces d'exécution ou vérifications *a posteriori*) lisibles par un humain des formules affichées, répondant à une requête fréquente des utilisateurs du calcul formel, qui manquent de moyens de juger quelle confiance accorder aux résultats.

Enfin et peut-être surtout, DynaMoW prend en charge l'aspect interactif de la navigation sur le site web. Les actions de l'utilisateur peuvent conduire à lancer des calculs supplémentaires dans les systèmes symboliques — donc côté serveur —, à rafraîchir en conséquence tout ou partie de la page courante, ou encore à charger une nouvelle page produite avec des paramètres qui dépendent de l'état de la navigation. Il est souhaitable aussi que les états des pages issus de ces calculs (ou plutôt, certains états judicieux) répondent à des adresses stables. Pour toutes ces raisons, le code DynaMoW est organisé en *services* dont les résultats, qui mêlent fragments de documents et valeurs symboliques, peuvent être affichés séparément, combinés en une seule page ou utilisés dans un calcul ultérieur.

Le court extrait du code source du DDMF présenté en figure 2.2 illustre quelques-uns de ces aspects. Ce n'est pas le lieu de s'étendre sur les choix de conception, l'architecture ou le mode d'utilisation de DynaMoW. Le lecteur pourra se reporter au rapport d'expérience de Chyzak et Darrasse [62] pour plus d'informations, notamment, sur la manière dont sont embarqués dans le programme maître des fragments de code écrits dans les langages natifs des calculateurs symboliques, ainsi que sur le modèle de services modulaires mentionné plus haut.

2.4 Futur

Le DDMF est encore en plein développement. Bien des informations restent à ajouter avant d'épuiser les résultats intéressants qui peuvent être calculés automatiquement. En ce qui concerne le contenu des pages existantes, citons :

- plus de développements sur des bases de fonctions variées, ainsi que de transformées intégrales ;
- l'ajout de code numérique généré automatiquement et téléchargeable, sur la base de travaux en cours avec Sylvain Chevillard ;
- un meilleur traitement des coupures (*branch cuts*) ;
- l'ajout d'informations sur les zéros des fonctions, notamment leur évaluation numérique, qui pourrait s'appuyer sur l'algorithme *bit-burst* (§8.5) combiné à la méthode de Newton et à des techniques de validation apparentées à celles du chapitre 9 ;

et, sur le plan de la classe de fonctions couvertes :

- le traitement des fonctions à paramètres comme les fonctions de Bessel, spécialisées en une valeur du paramètre donnée par l'utilisateur, mais aussi *via* une discussion automatique des résultats en fonction du paramètre [172] ;
- la possibilité pour l'utilisateur d'ajouter lui-même des fonctions.

Chapitre 3

Préliminaires : équations différentielles et récurrences linéaires

Ce chapitre collecte notations, définitions, et résultats de base utilisés dans la partie « théorique » du mémoire. La plupart sont classiques. L'index placé en fin de volume devrait permettre au besoin de les retrouver au fil de la lecture. Les rappels relatifs aux points singuliers des équations différentielles ne se trouvent pas ici mais au chapitre 4 ; ceux d'algorithmique, au chapitre 7.

3.1 Notations et conventions

3.1.1 Structures algébriques usuelles

Anneaux et corps. Les anneaux et corps sont sauf indication contraire de caractéristique nulle. La clôture algébrique d'un corps K est notée $\bar{K}$.

Séries. Si A est un anneau commutatif, on note $A[[z]]$ l'anneau des *séries formelles* en une indéterminée à coefficients dans A. Si K est un corps, on note $K((z))$ le corps des *séries de Laurent formelles* à coefficients dans K. Rappelons que les séries de Laurent formelles, de la forme $\sum_{n \geqslant -N} y_n\, z^n$, n'ont qu'un nombre fini de termes négatifs, et diffèrent en cela des séries de Laurent « des analystes » $\sum_{n=-\infty}^{\infty} y_n\, z^n$. Cette restriction permet de définir le produit sans avoir à associer de somme à des séries infinies. Les deux types de séries de Laurent interviennent dans ce mémoire.

L'anneau $A[z]$ des polynômes sur A est un sous-anneau de $A[[z]]$. L'anneau $K[[z]]$ est un sous-anneau de $K((z))$, et le corps $K(z)$ des fractions rationnelles sur K est un sous-corps de $K((z))$. Nous utiliserons souvent ces inclusions de façon implicite, par exemple pour extraire un coefficient des développements en série de fractions rationnelles.

Pour toute série $y(z)$ de la forme

$$y(z) = \sum_{k=-\infty}^{\infty} y_k\, z^k,$$

on note $[z^k]\, y = y_k$ le coefficient de z^k dans y. Les notations suivantes, empruntées à van der Hoeven [240], représentent des « tranches » de la série $y(z)$:

$$y_{m;n}(z) = \sum_{k=m}^{n-1} y_k\, z^k \qquad y_{m;}(z) = \sum_{k=m}^{\infty} y_k\, z^k \qquad y_{;n}(z) = \sum_{k=-\infty}^{n-1} y_k\, z^k.$$

Cela s'applique en particulier aux polynômes et fractions rationnelles.

Polynômes tordus et opérateurs linéaires. Toujours si A est un anneau commutatif, on note $A\langle\chi\rangle$ l'anneau des polynômes sur A en une indéterminée χ qui ne commute pas nécessairement avec les éléments de A. La relation de commutation entre χ et les éléments de A fait partie de la donnée de χ. Les éléments de $A\langle\chi\rangle$ sont appelés *polynômes tordus*. Le plus souvent, A est lui-même un anneau de polynômes, de fractions rationnelles ou de séries, et χ s'identifie à un opérateur sur A dont l'action

spécifie implicitement la règle de commutation. Je renvoie à la thèse de Chyzak [59, chap. I] pour une présentation plus soigneuse.

En particulier, si K est un corps, on note $K(z)\langle\partial\rangle$ l'anneau de polynômes tordus à coefficients dans $K(z)$ en une indéterminée ∂ sujette aux règles de commutation

$$\partial\, a = a\,\partial \quad (a \in K), \qquad \partial\, z = z\,\partial + 1. \tag{3.1}$$

La notation $K[z]\langle\partial\rangle$ désigne le sous-anneau de $K(z)\langle\partial\rangle$ formé des opérateurs à coefficients polynomiaux. Les éléments de $K[z]\langle\partial\rangle$ s'identifient naturellement aux opérateurs différentiels à coefficients polynomiaux sur $K[z]$, $K(z)$, $K[[z]]$ ou toute autre K-algèbre différentielle[1] contenant $K[z]$ *via* l'action à gauche de $K[z]\langle\partial\rangle$ définie par $z \cdot f = z\, f$ et $\partial \cdot f = f'$. Les égalités (3.1) traduisent la linéarité de la dérivation et la règle de Leibniz $(z\, f)' = z\, f' + f$. Nous confondrons le plus souvent l'indéterminée ∂ et son action, l'opérateur de dérivation par rapport à z, en écrivant au besoin ∂_z au lieu de ∂ s'il faut distinguer plusieurs dérivations.

On pose $\theta = z\,\partial$. Cet opérateur est appelé *dérivation d'Euler*. C'est une dérivation — on a $\theta \cdot (u\, v) = (\theta \cdot u)\, v + u\, (\theta \cdot v))$ — qui présente l'intérêt de ne pas changer le degré des polynômes ni la valuation des séries auxquels on l'applique. Le formulaire en figure 3.1 regroupe quelques règles de commutation et autres formules utiles dans les calculs sur les opérateurs différentiels usuels.

De même, on définit des anneaux $K(n)\langle S\rangle$ et $K[n]\langle S\rangle$, dont les indéterminées obéissent aux relations de commutation

$$S\, a = a\, S \quad (a \in K), \qquad S\, n = (n+1)\, S. \tag{3.2}$$

Ici, S s'interprète comme l'opérateur de *décalage* de la variable n. Lorsque N est une partie de $\mathbb{C}$ close par $\nu \mapsto \nu + 1$ (concrètement, $N = \mathbb{N}$, $N = \mathbb{Z}$ ou $N = \lambda + \mathbb{Z}$), les éléments de $K[n]\langle S\rangle$ agissent sur les suites indexées par N d'éléments de K par $(n \cdot u)_n = n\, u_n$ et $(S \cdot u)_n = u_{n+1}$. À nouveau, on confond ces polynômes avec les *opérateurs de récurrence* correspondants sur K^N.

Quand cela a une importance, on convient que les éléments d'un anneau $A\langle\chi\rangle$ sont écrits (sauf mention contraire) sous la forme $L = a_d\, \chi^d + a_{d-1}\, \chi^{d-1} + \cdots + a_0$, autrement dit avec les coefficients $a_i \in A$ à *gauche* de χ. Comme pour les polynômes usuels, on note alors $[\chi^k]\, L = a_k$.

Les opérateurs différentiels linéaires seront le plus souvent notés D ou L, leur ordre noté r, leurs coefficients — polynomiaux, rationnels ou séries suivant le contexte — notés a. Les opérateurs de récurrence seront généralement d'ordre s, notés P, Q, ou R, et à coefficients polynomiaux ou rationnels notés b.

$$\partial\, z^k = z^k\,\partial + k\, z^{k-1} \qquad \theta = z\,\partial \qquad \theta\, z = z\,(\theta + 1)$$
$$\partial^j\, z = z\,\partial^j + j\,\partial^{j-1} \qquad \theta^{\downarrow k} = z^k\,\partial^k \qquad z\,\theta = (\theta - 1)\, z$$
$$\partial\, z^{-1} = z^{-1}\,\partial - z^{-2} \qquad \partial\,\theta = \theta\,\partial + \partial \qquad \theta\, z^{-1} = z^{-1}\,(\theta - 1)$$

$$\partial^j (z^k\, f) = \sum_{i=0}^{j} \binom{j}{i} (\partial^{j-i}\, z^k)\, (\partial^i\, f) \qquad \partial^j\, z^k = \sum_{j-k \leqslant i \leqslant j} \binom{j}{i} k^{\downarrow(j-i)}\, z^{k-(j-i)}\, \partial^i$$

$\theta^j = \sum_{i=0}^{j} \left\{ {j \atop i} \right\} z^i\, \partial^i$, où les $\left\{ {i \atop j} \right\}$ sont les nombres de Stirling de seconde espèce

(OEIS A008277 [220])

Figure 3.1. Formulaire.

Matrices. Si A est un anneau, non nécessairement commutatif, on désigne par $A^{r \times s}$ le A-module des matrices $r \times s$ à coefficients dans A, et par $A^{s \times s}$ l'algèbre des matrices carrées de taille s à coefficients dans A.

1. Une algèbre différentielle est une algèbre munie d'une *dérivation*, c'est-à-dire une application linéaire $y \mapsto y'$ vérifiant la règle de Leibniz $(u\, v)' = u'\, v + u\, v'$.

3.1.2 Autres notations

Polynômes et fractions rationnelles. Si $P \in K[x]$ et $\zeta \in \bar{K}$, on note $\nu(\zeta, P)$ la *multiplicité* de ζ comme racine de P. En particulier, $\nu(\zeta, P) = 0$ si et seulement si $P(\zeta) \neq 0$. Le *polynôme réciproque* de P est noté

$$\operatorname{Rcpq} P = X^{\deg P} P(X^{-1}).$$

Lorsque f est une fraction rationnelle (irréductible), on note $\operatorname{num} f$ son numérateur et $\operatorname{den} f$ son dénominateur.

Logarithme. Les notations log ou ln désignent le logarithme complexe standard, c'est-à-dire la fonction analytique sur le plan fendu $\mathbb{C} \setminus \mathbb{R}_-$ et qui coïncide sur les réels strictement positifs avec le logarithme népérien réel, prolongée à $\mathbb{R}_-^*$ par

$$\log x = \lim_{\substack{\varepsilon \to 0 \\ \varepsilon > 0}} \log(x + i\,\varepsilon) \qquad (x < 0).$$

Les analyses d'algorithmes et de structures discrètes utilisent plutôt le logarithme binaire, noté lg, ou $\log_2$ quand on veut insister sur la base.

Asymptotique. Les notations asymptotiques $O(\cdot)$, $o(\cdot)$, $\Omega(\cdot)$ et $\Theta(\cdot)$ ont leur sens habituel en informatique. Soient f et g sont des fonctions définies au voisinage de ℓ, à valeurs dans un espace vectoriel normé. On écrit $f(x) = O(g(x))$ quand $x \to \ell$ si et seulement s'il existe $C > 0$ telle que $\|f(x)\| \leqslant C\,\|g(x)\|$ pour x suffisamment proche de ℓ. En présence de plusieurs variables, une assertion comme

$$f(n, m) = O(g(n, m)) \text{ quand } n, m \to \infty \text{ avec } m = O(n)$$

signifie qu'il existe C_1, C_2 tels que $\|f(n, m)\| \leqslant C_2\,\|g(n, m)\|$ quand $(n, m) \to (\infty, \infty)$ en restant confiné à $\{(n, m) | m \leqslant C_1\, n\}$.

La définition de $o(\cdot)$ est similaire en remplaçant la constante C par une fonction de x qui tend vers 0 en ℓ. J'écris parfois $f \ll g$, voire $f(x) \ll g(x)$, pour $f(x) = o(g(x))$. Le $\Omega(\cdot)$ est celui de Knuth [144] : on a $f(x) = \Omega(g(x))$ quand $x \to \ell$ si et seulement s'il existe $C > 0$ telle que $f(x) \geqslant C\,g(x)$ pour x proche de ℓ, et $f(x) = \Theta(g(x))$ si et seulement s'il existe $C, C' > 0$ telles que $C\,g(x) \leqslant f(x) \leqslant C'\,g(x)$ pour x proche de ℓ.

La notation $\tilde{O}(\cdot)$, analogue à $O(\cdot)$, cache des facteurs logarithmiques au lieu d'une constante : on note $f(x) = \tilde{O}(g(x))$ quand $x \to \ell$ si et seulement s'il existe k tel que

$$f(x) = O\Big(g(x)\,\big(\log g(x)\big)^k\Big)$$

quand $x \to \ell$.

Les notations $O(\cdot)$ et $\tilde{O}(\cdot)$ servent aussi à indiquer la troncature des séries formelles : on donne les premiers coefficients de $y \in \mathbb{C}[[z]]$ par

$$y(z) = y_0 + y_1\, z + \cdots + y_{n-1}\, z^{n-1} + O(z^n),$$

et (de façon moins classique) on écrit de même

$$y(z) = p_0(\log z) + p_1(\log z)\, z + \cdots + p_{n-1}(\log z)\, z^{n-1} + \tilde{O}(z^n)$$

lorsque $y(z) \in \mathbb{C}[\log z][[z]]$.

Divers.

- Le symbole $\propto$ n'est pas une notation asymptotique, mais signifie simplement « est proportionnel à ».
- On note

  $$\mathcal{D}(a, \rho) = \{z \in \mathbb{C} : |z - a| < \rho\} \quad \text{et} \quad \bar{\mathcal{D}}(a, \rho) = \{z \in \mathbb{C} : |z - a| \leqslant \rho\}$$

  les disques complexes ouvert et fermé centrés en a et de rayon ρ.

– Quand f est une fonction à valeurs dans un groupe multiplicatif, la notation f^k désigne la k-ième *puissance* de f, soit $f^k : x \mapsto f(x)^k$. En particulier,

$$\log^k x = (\log x)^k.$$

Les *itérés* sont quant à eux notés $f^{\circ k}(x) = f(f(\cdots(f(x))))$.

– Un exposant entier précédé d'une flèche représente une *factorielle montante* ou descendante :

$$x^{\uparrow k} = x\,(x+1)\cdots(x+k-1)$$
$$x^{\downarrow k} = x\,(x-1)\cdots(x-k+1).$$

– On note $\mathbb{1}[\cdot]$ le crochet d'Iverson, c'est-à-dire que pour une proposition P quelconque, l'expression $\mathbb{1}[P]$ vaut 1 lorsque P est vraie, et 0 sinon.

– Les doubles crochets servent à noter les intervalles d'entiers :

$$[\![i, j]\!] = \{i, i+1, ..., j\} \qquad (i, j \in \mathbb{Z} \cup \{\pm\infty\}).$$

– Dans une énumération notée par des points de suspension, un accent circonflexe au-dessus d'un terme signifie l'omission de ce terme : on écrit par exemple

$$a_1\, a_2 \cdots \hat{a_i} \cdots a_n = \prod_{\substack{1 \leqslant j \leqslant n \\ j \neq i}} a_j$$

ou encore $(a_i, ..., \hat{a_j}, ..., a_k) = (a_i, ..., a_{j-1}, a_{j+1}, ..., a_k)$.

– Dans les algorithmes, on distingue la *définition* d'un nom et l'*affectation* d'une valeur à une variable. L'instruction $a := b$ donne à a la valeur b, qui ne peut pas être modifiée par la suite. L'instruction $a \leftarrow b$ place la valeur b dans la variable modifiable a. Je privilégie lorsque cela n'alourdit pas trop le pseudo-code un style d'écriture des boucles où l'on définit un nouveau nom a_k à chaque itération plutôt qu'écraser le contenu d'une unique variable a.

3.2 Généralités sur les récurrences et équations différentielles linéaires

3.2.1 Récurrences

Soit $\mathbb{K}$ un corps, et soit N une partie de $\mathbb{C}$ close par successeur. Une récurrence linéaire homogène, d'ordre s, à valeurs dans $\mathbb{K}$, d'ensemble d'indices N est une équation de la forme

$$b_s(n)\, u_{n+s} + \cdots + b_1(n)\, u_{n+1} + b_0(n)\, u_n = 0, \tag{3.3}$$

où les $b_k : N \to \mathbb{K}$ sont des fonctions quelconques. En posant $R = b_s\, S^s + \cdots + b_1\, S + b_0$, où S désigne comme d'habitude l'opérateur de décalage, on écrit aussi (3.3) sous la forme compacte $R \cdot u = 0$.

CONVENTION 3.1. Les récurrences considérées dans ce manuscrit seront presque toujours linéaires et homogènes, et nous sous-entendrons souvent ces qualificatifs. ♢

Une *solution* de la récurrence (3.3) est une suite $(u_n)_{n \in N}$ à valeurs dans $\mathbb{K}$ dont les coefficients satisfont (3.3) pour tout $n \in N$. On omet souvent de préciser l'ensemble des indices. Celui-ci revêt cependant une certaine importance à plusieurs reprises dans ce document. Nous aurons notamment à distinguer entre les solutions indexées par les entiers naturels et les solutions indexées par les entiers relatifs, nulles aux indices négatifs. Par exemple, la suite $(n!)_{n \in \mathbb{N}}$ est solution de $u_{n+1} = (n+1)\, u_n$, alors que toute solution $(u_n)_{n \in \mathbb{Z}}$ s'annule pour $n \geqslant 0$. En revanche, le prolongement $(u_n)_{n \in \mathbb{Z}}$ de la factorielle défini par

$$u_n = \begin{cases} n!, & n \geqslant 0 \\ 0, & n < 0 \end{cases}$$

satisfait la récurrence $(n+1)^2\, u_{n+1} = (n+1)\, u_n$. Nous rencontrerons comme ensembles d'indices $N = \mathbb{N}$, $N = \mathbb{Z}$, et plus généralement $N = \lambda + \mathbb{N}$ ou $N = \lambda + \mathbb{Z}$ avec $\lambda \in \mathbb{C}$.

Convention 3.2. En l'absence d'indication contraire, les suites considérées dans ce mémoire sont à valeurs complexes et indexées par $\mathbb{Z}$. Nous les voyons aussi comme des fonctions de $\mathbb{C}$ dans $\mathbb{C}$ nulles sur $\mathbb{C} \setminus \mathbb{Z}$.

En particulier, les suites de coefficients de polynômes ou de séries sont toujours implicitement prolongées aux entiers relatifs par des zéros. ♢

Définition 3.3. On appelle *singularités de tête* (ou simplement singularités) de (3.3), ou de l'opérateur R, les indices $n \in N$ tels que $b_s(n-s) = 0$. On appelle *singularités de queue* ceux tels que $b_0(n) = 0$. Une récurrence (ou un opérateur) sans singularité de tête est dite *non singulière* ; une récurrence sans singularité de queue est dite *réversible*. ♢

En l'absence de singularité de tête en $\nu \in s + N$, la récurrence (3.3) détermine u_ν en fonction des $u_{\nu-k}$ pour $k \geqslant 0$. Supposons en revanche ν singularité de tête. Deux phénomènes se produisent alors simultanément :

(i) la relation (3.3) prise en $n = \nu - s$ impose une contrainte linéaire sur les $u_{\nu-k}$;

(ii) pourvu que celle-ci soit satisfaite, le coefficient u_ν peut être choisi indépendamment des $u_{\nu-k}$.

« En général », la contrainte (i) est non triviale, et fait chuter d'une unité la dimension de l'espace des suites définies jusqu'au rang $\nu - 1$ qui satisfont (3.3) pour $n = \nu - k$ où $k \geqslant 0$ par rapport à l'espace de celles qui ne vérifient l'équation qu'en $n = \nu - k$ avec $k > 0$. Inversement, comme le choix $u_{\nu-k} = 0$ satisfait toujours la contrainte exceptionnelle, on dispose d'une solution non triviale de support $\nu + \mathbb{N}$, à moins qu'une autre singularité dans $\nu + \mathbb{N}$ ne vienne changer la donne. (En particulier, une récurrence avec une singularité ν de partie réelle maximale admet une solution non triviale supportée par $\nu + \mathbb{N}$, unique à une constante multiplicative près.)

De même, suivant que ν est ou non singularité de queue, le coefficient u_ν est ou non déterminé en fonction de $u_{\nu+1}, u_{\nu+2}, \ldots$, avec éventuellement une contrainte linéaire sur ces derniers. On aboutit à la description suivante des espaces de solutions.

Proposition 3.4. Soient $E_{\mathbb{N}}$ et $E_{\mathbb{Z}}$ les espaces des solutions de (3.3) indexées respectivement par $\mathbb{N}$ et par $\mathbb{Z}$. Notons k le nombre de ses singularités de tête et k' celui de ses singularités de queue, tous deux supposés finis.

(a) Si la récurrence (3.3) est non singulière ($k = 0$), l'espace $E_{\mathbb{N}}$ a pour dimension s. Des conditions initiales $u_0, \ldots, u_{s-1}$ déterminent une unique solution $u \in E_{\mathbb{N}}$.

(b) Si la récurrence (3.3) est non singulière et réversible ($k = k' = 0$), l'espace $E_{\mathbb{Z}}$ a pour dimension s. La donnée de u_n en s indices consécutifs[2] quelconques détermine une unique solution $u \in E_{\mathbb{Z}}$.

(c) En présence de singularités, on a $s \leqslant \dim E_{\mathbb{N}} \leqslant s + k$. La donnée de u_n pour $n \in [\![0, s-1]\!] \cup \{\nu \geqslant s : b_s(\nu - s) = 0\}$ satisfaisant un certain système linéaire calculable explicitement détermine une unique solution $u \in E_{\mathbb{N}}$.

(d) De même, on a en général $s \leqslant \dim E_{\mathbb{Z}} \leqslant s + \min(k, k')$.

(e) Soit $Z = \{(u_n)_{n\in\mathbb{Z}} : \exists n_0 \in \mathbb{Z}, n < n_0 \Rightarrow u_n = 0\}$. Supposons $k \geqslant 1$. La dimension de l'espace $E_{\mathbb{Z}} \cap Z$ des solutions indexées par $\mathbb{Z}$ de (3.3) *nulles au voisinage de* $-\infty$ vérifie $1 \leqslant \dim(E_{\mathbb{Z}} \cap Z) \leqslant k$. La donnée de u_n pour $b_s(n-s) = 0$, satisfaisant un certain système linéaire calculable explicitement, détermine une unique solution $u \in E_{\mathbb{Z}} \cap Z$. □

2. Des valeurs de u_n en des indices n non consécutifs ne déterminent pas forcément de solution : par exemple, si $u_{n+2} + u_{n+1} + u_n = 0$, on a toujours $u_3 = u_0$, donc les indices 0 et 3 ne correspondent pas à des conditions initiales acceptables.

Quand l'ensemble des indices est stable par prédécesseur, on écrit aussi la récurrence (3.3) sous la forme équivalente

$$\tilde{b_0}(n)\,u_n + \tilde{b_1}(n)\,u_{n-1} + \cdots + \tilde{b_s}(n)\,u_{n-s} = 0, \qquad \tilde{b_k}(n) = b_{s-k}(n-s),$$

pour mettre en évidence les singularités de tête, voire sous d'autres formes décalées.

Nous rencontrerons enfin des « *récurrences non bornées* », ou « *récurrences d'ordre infini* », qui ne sont pas des récurrences au sens de cette section, mais des équations de la forme

$$b_0(n)\,u_n + b_1(n)\,u_{n-1} + \cdots = 0, \qquad \text{aussi écrites} \qquad \sum_{j=0}^{\infty} b_j(n)\,S^{-j}\cdot u.$$

Nous ne nous intéressons qu'à leurs solutions $(u_n)_{n\in\mathbb{Z}}$ nulles au voisinage de $-\infty$. La proposition 3.4(e) s'étend.

3.2.2 Équations différentielles à coefficients séries

Considérons maintenant l'équation différentielle linéaire homogène

$$a_r(z)\,y^{(r)} + \cdots + a_1(z)\,y'(z) + a_0(z)\,y(z) = 0, \tag{3.4}$$

dont nous laissons un instant dans le flou la nature des coefficients a_k. Je résume ici les propriétés des équations de ce type essentielles pour la suite, en renvoyant pour plus d'informations aux nombreux traitements détaillés des équations différentielles à coefficients analytiques [132, 203, 128, 127].

CONVENTION 3.5. Les équations différentielles considérées dans ce manuscrit sont en général linéaires et homogènes. Nous sous-entendrons souvent ces qualificatifs. ♢

DÉFINITION 3.6. Un point z où a_r s'annule est appelé *point singulier* (ou *singularité*) de l'équation (3.4). Un point qui n'est pas singulier est dit *ordinaire*[3]. ♢

Il est bien connu que si les a_k sont des fonctions analytiques, au voisinage de tout point ordinaire, l'équation (3.4) admet un espace vectoriel de dimension r de solutions elles-mêmes analytiques. Mais faisons un bref détour avant d'y revenir.

Solutions formelles. Nos raisonnements et calculs sur les équations différentielles sont bien souvent purement formels. Pour cette raison, il est commode de voir les équations différentielles à coefficients analytiques comme des cas particuliers d'équations à coefficients *séries formelles*, et les solutions analytiques comme des cas particuliers de solutions formelles. Cela se justifie par le fait que les séries formelles forment une algèbre différentielle, dont les séries convergentes sont une sous-algèbre.

Adoptons donc $a_k \in \mathbb{K}[[z]]$, où $\mathbb{K}$ est un sous-corps de $\mathbb{C}$. Posons

$$L(z,\partial) = a_r(z)\,\partial^r + \cdots + a_1(z)\,\partial + a_0(z) \in \mathbb{K}[[z]]\langle\partial\rangle,$$

de sorte que l'équation (3.4) s'écrit encore $L(z,\partial)\cdot y = 0$. Appliquée au point $z=0$, la définition 3.6 a un sens avec $a_k \in \mathbb{K}[[z]]$, et l'on peut donc parler de la nature ordinaire ou singulière de l'origine.

PROPOSITION 3.7.

(a) Une série formelle $y(z) = \sum_n y_n\,z^n$ est solution de l'équation $L(z,\partial)\cdot y = 0$ si et seulement si la suite (nulle pour $n<0$) de ses coefficients satisfait la récurrence d'ordre infini

$$L\Big(S^{-1}, (n+1)\,S\Big)\cdot (y_n)_{n\in\mathbb{Z}} = 0 \tag{3.5}$$

3. Et non régulier : les points singuliers sont eux-mêmes traditionnellement classifiés en *réguliers* et *irréguliers* ; voir le chapitre suivant.

c'est-à-dire

$$Q_0(n)y_n + Q_1(n)\, y_{n-1} + \cdots + Q_n(n)\, y_0 = 0$$

pour certains polynômes $Q_0, Q_1, \ldots \in \mathbb{K}[n]$.

(b) Supposons que l'origine est un point ordinaire de (3.4). Pour tous $\ell_0, \ldots, \ell_{r-1}$, l'équation (3.4) admet alors une unique solution $y \in \mathbb{K}[[z]]$ telle que $y^{(k)}(0) = \ell_k$ pour $0 \leqslant k \leqslant r-1$. ♢

Démonstration (esquisse). L'assertion (a) s'obtient par exemple en injectant dans l'équation différentielle une série à coefficients indéterminés et en étudiant l'action sur les monômes des opérateurs z et ∂ ; l'assertion (b), en considérant le système différentiel du premier ordre compagnon de l'équation et la récurrence associée. Nous prouverons des versions plus générales de l'une et l'autre au chapitre suivant (proposition 4.14 et remarque 4.22). □

Multiplier à gauche par une puissance de z l'équation différentielle (3.4), ou par une puissance de S la récurrence (3.5), n'en change pas les solutions. Il s'avère souvent commode d'écrire

$$L(z, \partial) = z^{-\nu} \tilde{L}(z, \theta), \qquad \theta = z\,\partial,$$

pour un ν convenable. On a alors

$$L(z,\partial)\cdot y = 0 \quad \Leftrightarrow \quad \tilde{L}(z,\theta)\cdot y = 0 \quad \Leftrightarrow \quad \tilde{L}(S^{-1}, n)\cdot (y_n)_{n\in\mathbb{Z}} = 0. \tag{3.6}$$

Les substitutions

$$z \mapsto S^{-1}, \quad \partial \mapsto (n+1)\, S, \quad \theta = z\,\partial \mapsto n$$

qui font passer de l'équation différentielle (3.4) à la récurrence d'ordre infini (3.5) définissent des morphismes de $\mathbb{K}$-algèbres entre les espaces de séries non commutatives correspondants.

Le point (b) de la proposition 3.7 établit que l'espace des solutions formelles dans $\mathbb{K}[[z]]$ d'une équation différentielle d'ordre r à coefficients dans $\mathbb{K}[[z]]$ dont l'origine est un point ordinaire est de dimension r. Il n'y a pas équivalence. Un point singulier où l'équation admet néanmoins un espace de solutions séries de dimension r s'appelle une *singularité apparente*. Nous adopterons comme *base canonique* de solutions formelles en un point la famille des solutions telles que

$$\begin{cases} y_k = \dfrac{1}{k!}\, y^{(k)}(0) = 1 \\ y_j = 0, \qquad\qquad\qquad j \in [\![0, r-1]\!] \setminus \{k\} \end{cases}$$

pour $k = 0, \ldots, r-1$.

Solutions analytiques. Lorsque les séries a_k sont convergentes — et donc, définissent des fonctions analytiques sur un voisinage de l'origine — les solutions formelles de (3.4) aussi. Précisément, si 0 est un point ordinaire de (3.4), et si les séries a_k sont convergentes sur le disque $\mathcal{D}(0,\ \rho)$, alors toute solution formelle de (3.4) converge sur $\mathcal{D}(0, \rho)$. On aboutit finalement au classique théorème de Cauchy pour les équations différentielles linéaires (voir par exemple Poole [203, §2]).

Théorème 3.8. Au voisinage d'un point ordinaire, une équation différentielle à coefficients analytiques admet un espace vectoriel de solutions analytiques de dimension égale à son ordre. □

Soit $\boldsymbol{S}$ l'ensemble des points singuliers de (3.4), que nous supposons isolés dans $\mathbb{C}$. Les solutions (3.4) admettent un *prolongement analytique* le long de tout arc polygonal tracé dans $\mathbb{C} \setminus \boldsymbol{S}$ (voir Hille [128, §9.2–9.3]). Le prolongement analytique d'une solution à partir d'un point-base ζ fixé au voisinage duquel elle est analytique associe à chaque arc basé en ζ une valeur qui ne dépend que de la classe d'homotopie de

celui-ci dans $\mathbb{C} \setminus \boldsymbol{S}$. Une manière de construire le prolongement consiste simplement à recouvrir l'arc de disques sur lesquels le théorème précédent s'applique.

COROLLAIRE 3.9. Soit U un ouvert simplement connexe de $\mathbb{C} \setminus \boldsymbol{S}$. L'équation (3.4) admet un espace vectoriel de dimension r de solutions analytiques sur U. Une solution y est déterminée de façon unique par des conditions initiales $y(\zeta), ..., y^{(r-1)}(\zeta)$ en un point $\zeta \in U$ quelconque. □

Moyennant la définition des fonctions analytiques et du prolongement analytique sur une surface de Riemann [221, 215], on peut remplacer U par le revêtement universel de $\mathbb{C}\setminus\boldsymbol{S}$. Nous nous en tiendrons à la vision naïve suivant laquelle une solution admet une valeur « à l'extrémité de tout chemin ».

3.3 D-finitude

3.3.1 Définitions

Nous avons déjà vu que les objets de notre étude, les fonctions D-finies, sont les solutions d'équations différentielles linéaires à coefficients *polynomiaux*, et le chapitre 1 a été l'occasion d'illustrer leurs propriétés essentielles. Énonçons-les (un peu) plus formellement. La théorie résumée ici est, à de petites nuances près, celle développée dans l'article de Stanley [222], auquel je renvoie pour plus de détails.

DÉFINITION 3.10. Une série formelle y est dite D-finie sur un corps $\mathbb{K}$ si elle satisfait une équation différentielle linéaire

$$a_r(z)\, y^{(r)}(z) + \cdots + a_1(z)\, y'(z) + a_0(z)\, y(z) = 0 \tag{3.7}$$

à coefficients $a_k \in \mathbb{K}[z]$. ♢

De façon équivalente, une série $y \in \mathbb{C}((z))$ est D-finie sur $\mathbb{K} \subset \mathbb{C}$ si le sous-$\mathbb{K}(z)$-espace vectoriel de $\mathbb{C}((z))$ engendré par $y, y', y'', ...$ est de dimension finie.

Nous n'imposons pas à y d'être à coefficients dans $\mathbb{K}$. Nous rencontrerons notamment des séries $y \in \mathbb{C}[[z]]$ D-finies sur $\mathbb{Q}$ sans être à coefficients rationnels.

Si l'origine est un point ordinaire de (3.7), la série y peut être représentée par cette équation accompagnée de conditions initiales $y(0), y'(0), ..., y^{(r-1)}(0)$. Naturellement, une série D-finie sur $\mathbb{K}$ ainsi donnée avec des conditions initiales elles-mêmes dans $\mathbb{K}$ appartient à $\mathbb{K}[[z]]$.

DÉFINITION 3.11. Une fonction analytique est dite D-finie sur un corps $\mathbb{K} \subset \mathbb{C}$ si elle est solution d'une équation différentielle linéaire à coefficients dans $\mathbb{K}[z]$. ♢

Le développement de Taylor en un point de $\mathbb{K}$ d'une fonction D-finie sur $\mathbb{K}$ est une série D-finie sur $\mathbb{K}$.

Une équation différentielle à coefficients polynomiaux n'a qu'un nombre fini de points singuliers. D'après le corollaire 3.9, une fonction D-finie y admet un prolongement analytique à tout ouvert du plan complexe évitant les points singuliers de l'équation qui la définit, voire d'après la remarque qui suit au revêtement universel de $\mathbb{C}$ privé d'un nombre fini de points. Une singularité de y se situe nécessairement en un point singulier de l'équation, et la fonction y peut ou non présenter une singularité en un point singulier donné.

Les séries D-finies de $\mathbb{K}[[z]]$ ont un pendant dans l'algèbre (isomorphe à $\mathbb{K}[[z]]$ en tant qu'espace vectoriel) $\mathbb{K}^{\mathbb{N}}$: les suites P-récursives.

DÉFINITION 3.12. Soit N une partie de $\mathbb{C}$ close par successeur. Une suite $(u_n)_{n\in N}$ est dite P-récursive sur un corps $\mathbb{K}$ si elle est solution d'une récurrence

$$b_s(n)\, u_{n+s} + \cdots + b_1(n)\, u_{n+1} + b_0(n)\, u_n = 0$$

à coefficients polynomiaux $b_k \in \mathbb{K}[n]$. ♢

Convention 3.13. Quand une série formelle ou une fonction D-finie y intervient dans un algorithme, il est sous-entendu (sauf indication contraire) que y est codée par une équation différentielle d'ordre r dont l'origine est un point ordinaire et des conditions initiales $y(0), y'(0), ..., y^{(r-1)}(0)$. Quand une suite P-récursive intervient dans un algorithme, il est sous-entendu qu'elle est codée par une récurrence et des conditions initiales en nombre et à des indices convenables. ♢

3.3.2 Équations différentielles et récurrences

Le lien essentiel entre fonctions D-finies et suites P-récursives est le suivant [222].

Proposition 3.14. Une série formelle est D-finie si et seulement si la suite de ses coefficients est P-récursive. □

L'expression « la suite de ses coefficients » contient une petite ambiguïté. Si l'on a

$$L(z,\partial)\cdot\sum_{n=0}^{\infty} y_n\, z^n = 0 \qquad \text{où} \quad L \in \mathbb{K}[z]\langle\partial\rangle,$$

l'opérateur $L(S^{-1}, (n+1)\,S)$ annule la suite $(y_n)_{n\in\mathbb{Z}}$ d'après la proposition 3.7(a), et donc en particulier la suite $(y_n)_{n\in\mathbb{N}}$. Comme l'équation différentielle est à coefficients polynomiaux, la récurrence associée est d'ordre fini. On dispose donc d'un algorithme simple de calcul de la récurrence associée à une équation différentielle.

Inversement, si $(y_n)_{n\in\mathbb{Z}}$ satisfait une récurrence de la forme $L(n,S)\cdot(y_n)_{n\in\mathbb{Z}} = 0$ avec $L(n,S) \in \mathbb{K}[n]\langle S\rangle$, on a $z^\nu\, L(\theta, z^{-1})\cdot\sum_n y_n\, z^n = 0$, et cette équation différentielle est à coefficients polynomiaux pour ν assez grand. En revanche, l'application de la même substitution à une récurrence écrite pour la suite $(y_n)_{n\in\mathbb{N}}$ peut fort bien conduire à une équation différentielle singulière à l'origine et sans solution série. Le sens « si » de la proposition 3.14 est tout de même vrai dans ce cas. Pour calculer l'équation différentielle associée à une récurrence, on commence par multiplier celle-ci par un polynôme de manière à introduire des singularités telles que la suite prolongée par zéro aux négatifs soit elle aussi solution (voir proposition 3.4). Cela donne lieu à l'algorithme 5.11 page 85, énoncé là où nous l'utilisons.

Notons qu'ainsi, dans le contexte des suites indexées par les entiers naturels, le passage d'une équation différentielle à une récurrence et celui d'une récurrence à une équation différentielle ne sont pas inverses l'un de l'autre. C'est à cette spécification que répondent les fonctions `diffeqtorec` et `rectodiffeq` de gfun. Pour les applications de ce mémoire, la correspondance « directe » (3.6) sera souvent appropriée.

La proposition 3.14 permet de calculer aisément les coefficients du développement de Taylor d'une fonction D-finie y en un point ordinaire quelconque de l'équation différentielle qui la définit. Elle a aussi la la conséquence suivante.

Corollaire 3.15. Soit y une série D-finie sur $\mathbb{K}$. Alors la suite $(y_{;n}(\zeta))_{n\in\mathbb{N}}$ des sommes partielles de y en un point $\zeta \in \mathbb{K}$ est P-récursive. Si en outre y converge en ζ, il en va de même de la suite $(y_{n;}(\zeta))_{n\in\mathbb{N}}$ des restes. ♢

Démonstration. Soit (y_n) la suite des coefficients de y. Soit $L(n,S) \in \mathbb{K}[n]\langle\partial\rangle$ tel que $L(n,S)\cdot(y_n)_{n\in\mathbb{Z}} = 0$. On a alors $L(n,\zeta^{-1}S)\cdot(y_n\,\zeta^n) = 0$, et donc

$$L(n,\zeta^{-1}S)\,(S-1)\cdot(y_{n;}(\zeta))_{n\in\mathbb{N}} = 0.$$

Les restes ne diffèrent des sommes partielles que par une constante additive. □

Il existe une seconde manière importante d'exprimer récursivement les coefficients de Taylor d'une fonction D-finie. Il s'agit de mettre l'équation différentielle sous forme « unitaire » en la divisant par son coefficient de tête. On obtient une équation différentielle à coefficients dans $\mathbb{K}(z)$, à laquelle la proposition 3.7 associe une récurrence,

cette fois-ci réellement *d'ordre infini* en général. Cette seconde récurrence est peu appropriée au calcul des coefficients, mais nous utiliserons ses propriétés théoriques.

3.3.3 Propriétés de clôture

Les classes des suites et fonctions D-finies sont stables par un certain nombre d'opérations usuelles [222, 214, 33].

PROPOSITION 3.16. Soit A une $\mathbb{K}$-algèbre. Les suites P-récursives sur $\mathbb{K}$ d'éléments de A forment une $\mathbb{K}$-algèbre. Les séries formelles D-finies sur $\mathbb{K}$ à coefficients dans A forment une $\mathbb{K}$-algèbre stable par dérivation, primitivation, et produit d'Hadamard. (Le produit d'Hadamard est l'opération qui, aux séries de termes généraux $u_n\, z^n$ et $v_n\, z^n$, associe celle de terme général $u_n\, v_n\, z^n$.) Il en va de même des fonctions analytiques D-finies sur $\mathbb{K}$. □

PROPOSITION 3.17. La composée $f \circ g$ d'une série (ou fonction analytique) D-finie f et d'une série (ou fonction) algébrique g telle que $g(0) = 0$ est D-finie. □

C'est vrai plus généralement lorsque la composée a un sens. En particulier, les fonctions algébriques sont D-finies, et une fonction D-finie le reste après un changement de variable rationnel comme (en vue du chapitre 8) $z \mapsto a + z$ quand a n'est pas une singularité, ou encore (en vue du chapitre 9) $z \mapsto \frac{1}{2}\,(z + z^{-1})$ quand il n'y a pas de singularité sur $[-1; 1]$.

L'inverse d'une série D-finie ou la composée de deux séries D-finies ne sont D-finies que dans des cas exceptionnels [33, §8.4].

La preuve de chacune des affirmations des propositions 3.16 et 3.17 donne lieu à un *algorithme* opérant sur la structure de données « équation + conditions initiales » pour réaliser l'opération correspondante. Ces algorithmes sont implémentés dans gfun, comme nous l'avons vu en §1.2.1.

PROPOSITION 3.18. Une solution d'équation différentielle inhomogène à coefficients dans $\mathbb{K}[z]$ dont le second membre est D-fini sur $\mathbb{K}$ est elle-même D-finie sur $\mathbb{K}$. □

Notamment, les solutions d'une équation différentielle linéaire à coefficients et second membre polynomiaux sont D-finies. Ces équations constituent une variante, parfois avantageuse (et gérée par gfun), de notre structure de données pour les fonctions D-finies. On peut explicitement *homogénéiser* une telle équation, c'est-à-dire calculer une équation homogène dont les solutions contiennent celles de l'équation de départ. Dans ce mémoire et dans NumGfun, nous nous limitons pour éviter de compliquer les algorithmes aux équations différentielles homogènes.

3.3.4 Intermède : (in)décidabilité

Les résultats suivants visent à donner une idée de « ce que l'on peut espérer » concernant le traitement algorithmique des fonctions D-finies, et, inversement, à situer les difficultés majeures. Ils ne sont presque pas utilisés explicitement dans la suite.

Un attrait essentiel de la classe des fonctions D-finies pour le calcul formel est, nous y avons déjà fait allusion, que *l'égalité y est décidable*. Il suffit en effet, pour tester si deux fonctions D-finies sont égales, de calculer une équation qui les annule toutes deux et de comparer des conditions initiales en nombre fini. Le théorème de Richardson témoigne qu'il s'agit d'une propriété assez remarquable.

THÉORÈME 3.19. [208] Le test à zéro des expressions obtenues à partir des rationnels, des constantes π et $\ln 2$, et d'une indéterminée par composition des opérations d'anneau et des fonctions sin, exp et $|\cdot|$ est indécidable. ♢

Hélas, si l'égalité des *fonctions* D-finies est décidable, on ignore ce qu'il en est de celle de leurs *valeurs*. La question suivante est réputée extrêmement difficile.

Question 3.20. Existe-t-il un algorithme qui, pour une fonction y spécifiée par une équation différentielle linéaire d'ordre r à coefficients dans $\mathbb{Q}[z]$, sans point singulier dans le disque fermé $\bar{\mathcal{D}}(0,1)$, et des conditions initiales $y(0), ..., y^{(r-1)}(0) \in \mathbb{Q}$, décide si $y(1) = 0$? ♢

Le simple problème de déterminer si une fonction réelle donnée par une équation différentielle linéaire à coefficients *constants* algébriques réels et des conditions initiales elles-mêmes algébriques réelles s'annule sur $\mathbb{R}_+$ est NP-difficile, et l'on ne sait pas s'il est décidable [16]. Dans le même ordre d'idée, le rayon de convergence d'une série D-finie est minoré trivialement par la distance à la singularité la plus proche de l'équation différentielle associée, mais on ignore s'il est calculable. Pour des équations différentielles non linéaires, de la forme $p(y, ..., y^{(r)}) = 0$ où $p \in \mathbb{Z}[x_0, ..., x_r]$, il est connu que ce n'est pas le cas [74, §4].

La situation est analogue du côté des suites P-récursives. Si l'égalité de deux suites est facile à tester, décider si une suite P-récursive s'annule contient la célèbre question ouverte suivante, appelée problème de (Pisot-)Skolem. On dispose d'algorithmes pour les récurrences d'ordre jusqu'à 5 [123], et, à nouveau, il est connu que le problème général est NP-difficile [29].

Question 3.21. Existe-t-il un algorithme qui, étant données une récurrence linéaire à coefficients constants entiers et des conditions initiales elles-mêmes entières, décide si la suite qu'elles définissent prend la valeur 0 ? ♢

En ce qui concerne les questions étudiées dans ce mémoire, ces difficultés restreignent sévèrement la possibilité, sans progrès majeurs :

- d'évaluer les fonctions D-finies à précision *relative* arbitraire (il faut pouvoir décider que le résultat est nul lorsque c'est le cas, au lieu d'augmenter indéfiniment la précision, voir aussi van der Hoeven [236]) ;
- de donner des *bornes fines* sur des suites P-récursives ou des fonctions D-finies, et notamment de tester la positivité d'une suite P-récursive arbitraire (par exemple, la non-nullité de $(u_n) \in \mathbb{Q}^{\mathbb{N}}$ équivaut à la positivité de $u_n^2 - 1$; voir aussi Gerhold [109]).

3.4 Asymptotique

3.4.1 Généralités

Les comportements que peuvent présenter les fonctions D-finies au voisinage de leurs singularités sont fortement contraints par des théorèmes généraux sur les solutions asymptotiques d'équation différentielles au voisinage de points singuliers. De même, satisfaire une récurrence à coefficients polynomiaux limite la gamme de comportements à l'infini possibles pour une suite complexe. Les uns et les autres sont reliés par le principe général, à la base de l'*analyse de singularité* [94], qui veut que le comportement d'une fonction analytique au voisinage de sa singularité de plus petit module non nul détermine le comportement à l'infini de la suite des coefficients de son développement en série à l'origine. L'avatar le plus simple de ce principe est la classique « règle de Cauchy » de convergence des séries entières.

Proposition 3.22. La série $\sum_{n=0}^{\infty} y_n z^n$ a pour rayon de convergence $\limsup_{n\to\infty} |y_n|^{1/n}$. □

Concernant les comportements possibles pour une fonction D-finie au voisinage d'une de ses singularités, on a le théorème général suivant, qui ne sera pas explicitement utilisé dans la suite. Je renvoie à Wasow [252] pour la preuve, ainsi qu'à van der Hoeven [239] et aux conférences [2] pour plus de références et un aperçu des développements ultérieurs. Rappelons que l'on dit que $f(x)$ est *asymptotique* à une

somme formelle $g(x)=\sum_{n=0}^{\infty} g_n(x)$, et l'on écrit $f(x)\sim g(x)$, lorsque pour tout $N\geqslant 0$, $f(x)=\sum_{n=0}^{N} g_n(x)+O(g_{n+1}(x))$.

THÉORÈME 3.23. Supposons que l'origine est un point singulier de l'équation différentielle à coefficients séries formelles (3.4). Alors celle-ci admet une base de solutions formelles $y_{[i]}(z)$ chacune de la forme

$$e^{P_i(z^{1/p_i})}\, z^{\alpha_i} \sum_{n=0}^{\infty} \sum_{k=0}^{m_i} c_{i,n,k}\, z^{n/p_i} (\log z)^k$$

où $p, m\in\mathbb{N}$, $\alpha\in\mathbb{C}$, $P\in\mathbb{C}[z]$, et les $c_{n,k}$ sont des nombres complexes. De plus, si les coefficients de l'équation sont convergents, toute solution analytique sur un secteur angulaire $\Delta=\{r\,e^{i\theta} : (r>0)\wedge(\theta_{\min}<\theta<\theta_{\max})\}$ d'ouverture $\theta_{\max}-\theta_{\min}$ assez petite est asymptotique à une combinaison linéaire des $y_{[k]}$ quand $z\to 0$ dans Δ. □

Nous ferons en revanche abondamment appel à la structure des solutions d'une équation différentielle au voisinage d'un point singulier *régulier*, détaillée au chapitre suivant.

3.4.2 Le théorème de Perron-Kreuser

Venons-en au cas des récurrences, et au principal résultat asymptotique que nous utiliserons véritablement. Supposons que les coefficients $b_k(n)$ de l'équation

$$b_s(n)\, u_{n+s} + b_{s-1}(n)\, u_{n+s-1} + \cdots + b_{s'}(n)\, u_{n+s'} = 0 \tag{3.8}$$

(où s' peut être négatif) ont des comportements asymptotiques de la forme

$$\forall k, \quad b_k(n)\sim c_k\, n^{d_k} \qquad \text{quand } n\to\infty$$

avec $c_k\in\mathbb{C}$ et $d_k\in\mathbb{Z}$. C'est le cas si ce sont des fractions rationnelles en n. Supposons de plus que (u_n) est une solution telle que

$$\frac{u_{n+1}}{u_n}\sim \lambda\, n^{\kappa} \qquad \text{quand } n\to\infty.$$

Pour que la relation (3.8) soit vérifiée quand $n\to\infty$, il est nécessaire que les termes asymptotiquement dominants se compensent, et donc que l'exposant $d_k+\kappa\, k$ le plus grand soit atteint au moins deux fois.

Cela entraîne que $-\kappa$ doit être parmi les pentes des arêtes d'un diagramme appelé *polygone de Newton* de l'équation. Les figures 5.2 page 86 et 9.1 page 183 présentent des exemples de polygones de Newton.

DÉFINITION 3.24. Le polygone de Newton de (3.8) est l'enveloppe convexe supérieure des points $(k,d_k)\in\mathbb{R}^2$ où $s'\leqslant k\leqslant s$. Si $E=[A,B]$ désigne une arête du polygone de Newton, on note $\kappa(E)$ l'opposé de sa pente, et on définit l'équation caractéristique associée à E (ou à $\kappa(e)$) par

$$\chi_E(\lambda)=\sum_{(k,d_k)\in E} c_k\,\lambda^{k-t}=0$$

où $(t,d_t)=A$ est l'extrémité gauche de E. ♢

Observons que la somme des degrés des différentes équations caractéristiques est égale à l'ordre de la récurrence.

Le théorème suivant, dit de Perron-Kreuser [199, 151, 200], décrit la structure asymptotique des solutions d'une récurrence à partir de son polygone de Newton. Guelfond [120], Meschkowski [171] ou encore Milne-Thomson [177] en offrent des discussions plus complètes.

Théorème 3.25. Pour toute arête E du polygone de Newton de la récurrence (3.8), notons $\lambda_{E,1}, \lambda_{E,2}, \ldots$ les racines de χ_E, comptées avec multiplicité.

(a) Supposons que pour toute arête E, les modules $|\lambda_{E,i}|$ des racines de χ_E sont deux à deux distincts. Alors toute solution non ultimement nulle de (3.8) satisfait
$$\frac{u_{n+1}}{u_n} \sim \lambda_{E,i}\, n^{\kappa(E)} \qquad \text{quand } n \to \infty$$
pour une certaine arête E et un certain i.

(b) Si en outre (3.8) est réversible, elle admet une base de solutions
$$(u_n^{[E,i]})_{E, 1 \leqslant i \leqslant \deg \chi_E}$$
telle que
$$\frac{u_{n+1}^{[E,i]}}{u_n^{[E,i]}} \sim \lambda_{E,i}\, n^{\kappa(E)} \qquad \text{quand } n \to \infty.$$

(c) Dans le cas où il existe E et $i \neq j$ avec $|\lambda_{E,i}| = |\lambda_{E,j}|$, les analogues des deux assertions précédentes tiennent mais avec la conclusion plus faible
$$\limsup_{n\to\infty} \left| \frac{u_n^{[E,i]}}{n!^{\kappa(E)}} \right|^{1/n} = |\lambda_{E,i}|. \qquad \diamondsuit$$

Le point (a) du théorème 3.25 s'appelle aussi théorème de Poincaré [202]. Différentes extensions et raffinements dans des cas particuliers de ce théorème sont disponibles dans la littérature (voir par exemple Schäfke [216], Noble [193] et les références qu'ils mentionnent). Certains des résultats de ce mémoire peuvent être affinés en conséquence lorsque ces résultats plus précis s'appliquent.

3.4.3 Développements asymptotiques de suites P-récursives

En fait, il est au moins conjecturé [94, §VIII.7] que toute suite P-récursive u admet un développement asymptotique de la forme
$$u_n \underset{n\to\infty}{\sim} \sum_{j}^{(\text{finie})} \lambda_j\, n!^{\kappa_j}\, e^{P_j(\sqrt[m]{n})}\, n^{K_j} \sum_{k=0}^{\infty} \sum_{\ell=0}^{L} c_{k,\ell}\, n^{\mu k} \log^{\ell} n \tag{3.9}$$
où les coefficients λ_j de la combinaison linéaire ainsi que les K_j sont des complexes, μ et les exposants κ_j de $n!$ des rationnels, m un entier positif, et les P_j des polynômes à coefficients complexes. Cela découle d'un résultat de Birkhoff et Trjitzinsky [28, 256, 194], qui demande seulement que les coefficients $b_j(n)$ de la récurrence admettent eux-mêmes un développement de la forme
$$b_j(n) \underset{n\to\infty}{\sim} n^{K/t} \sum_{k=0}^{\infty} c_k\, n^{-k/t}, \qquad K \in \mathbb{Z}, \quad t \in \mathbb{N}^*, \quad c_k \in \mathbb{C}.$$
Ce résultat a été contesté [130, p.128] (voir aussi les commentaires de Wimp [257, note 3], Odlyzko [194, §9.2], Flajolet et Sedgewick [94, p. 583]) ; cependant, il semble que les zones d'ombre qui pouvaient exister dans la preuve originale aient été éclairées depuis [131, 244, 161].

Dans le cas de récurrences à coefficients polynomiaux, les méthodes analytiques permettent souvent d'obtenir des développements de ce type sur des bases mieux établies, en « transférant » les conclusions des théorèmes 3.23 et 4.11 appliqués à la fonction génératrice $\sum_n u_n\, z^n$. En particulier, l'analyse de singularité démontre la formule (3.9) dès que les singularités de l'équation différentielle qui gouvernent l'asymptotique des coefficients des solutions sont des points singuliers réguliers [94, §VII.9.2]. La méthode du col permet d'aboutir au même type de conclusion au cas par cas dans une grande partie des autres situations qui apparaissent en pratique, mais sans fournir à ce jour d'énoncé complètement général [94, §VIII].

Pour éviter tant les terrains incertains que trop de distinctions de cas, nous nous cantonnons dans ce mémoire au niveau de précision du théorème de Perron-Kreuser. Le lecteur trouvera un panorama plus complet des approches de l'asymptotique des suites P-récursives dans la thèse récente de Noble [193].

Chapitre 4

Points singuliers réguliers

⟨Insérez ici la citation incontournable de *Sherlock Holmes*.⟩

Ce chapitre poursuit les rappels mathématiques relatifs aux fonctions D-finies. J'y reprends quelques résultats bien connus sur les solutions d'une équation différentielle au voisinage d'un point singulier, ainsi que les bases de la théorie classique des points singuliers dits réguliers. Ces derniers vont jouer un rôle important dans les chapitres 5 à 8. Je compare ensuite plusieurs variantes de la méthode de Frobenius, qui sert à calculer explicitement des développements en série de solutions au voisinage des points singuliers réguliers. Une observation semble nouvelle : celle que l'une de ces variantes, la méthode de Poole, peut se voir comme une généralisation directe du morphisme $z \mapsto S^{-1}$, $\theta \mapsto n$ employé au chapitre précédent pour passer d'une équation différentielle à une récurrence. J'ai déjà utilisé cette reformulation dans un article présentant NumGfun [174, §4], et une petite partie du texte du chapitre dérive de cet article.

4.1 Introduction

Contexte. Le théorème de Cauchy entraîne qu'une fonction D-finie est analytique au voisinage de tout point *ordinaire* de l'équation différentielle qui la spécifie. Il n'y a aucune raison qu'il en aille de même au voisinage d'un point singulier. Les expressions $e^{1/z}$ ou $\log z$, toutes deux D-finies, présentent des singularités essentielles en zéro, et ne sont analytiques que sur $\mathbb{C} \setminus \{0\}$ pour la première, ou sur des domaines comme $\mathbb{C} \setminus \mathbb{R}_-$ pour la seconde. Satisfaire une équation différentielle dont les coefficients ne peuvent avoir comme singularité qu'un pôle au point singulier considéré impose toutefois certaines contraintes ; et nous allons voir que les comportements de ces deux exemples sont en quelque sorte les pires possibles. Une fonction D-finie n'est « jamais très loin » d'être analytique, même au voisinage d'un point singulier.

Plus particulièrement, on isole une classe de points singuliers « modérés », dits *réguliers*, autour desquels existe une base de solutions de la forme

$$y(z) = z^\lambda \sum_{k=0}^{t-1} u_k(z) \, (\log z)^k. \tag{4.1}$$

pour divers $\lambda \in \mathbb{C}$, où les $u_k(z)$ sont des fonctions analytiques. L'existence de récurrences sur les coefficients, déjà vue pour les développements en série de fonctions D-finies en les points ordinaires, s'étend aussi. Un théorème classique dit en effet que si, dans l'expression ci-dessus, on pose

$$u_k(z) = \sum_{n=0}^{\infty} u_{k,n} \, z^n,$$

alors les $u_{k,n}$ satisfont un système de relations de la forme

$$\begin{cases} Q_0(n)\, u_{t-1,n} + \cdots + Q_s(n)\, u_{t-1,n-s} = 0 \\ Q_0(n)\, u_{t-2,n} + \cdots + Q_s(n)\, u_{t-2,n-s} = f_{t-2}(u_{t-1,n}, \ldots, u_{t-1,n-s}) \\ \qquad\qquad \vdots \\ Q_0(n)\, u_{0,n} + \cdots + Q_s(n)\, u_{0,n-s} = f_0(u_{t-1,n}, \ldots, u_{t-2,n}, \ldots, u_{1,n}, \ldots, u_{1,n-s}). \end{cases} \tag{4.2}$$

Les Q_j sont des polynômes. Les seconds membres dépendent linéairement de $u_{1,n}, \ldots, u_{2,n}, \ldots, u_{t-1,n}, \ldots, u_{t-1,n-s}$ et s'expriment à partir des dérivées successives des Q_j.

Nous nous intéressons ici à ces résultats dans l'optique du calcul effectif des séries u_k. Les récurrences (4.2) jouent un rôle central. La théorie se développe pratiquement à l'identique pour les équations différentielles à coefficients analytiques (au voisinage d'une singularité isolée) plutôt que polynomiaux, et nous allons souvent nous placer dans ce cadre. Comme dans le cas des développements aux points ordinaires (voir §3.2.2), il faut alors remplacer les récurrences (4.2) d'ordre borné s par des expressions pouvant faire intervenir tous les coefficients « précédents » des séries u_k.

Historique. Le développement de ces idées est relaté par Gray [117, Chap. II]. Sommairement, la construction d'une base de solutions (4.1) au voisinage d'un point singulier régulier, pour une équation scalaire à coefficients analytiques, est habituellement attribuée à Fuchs (1866) [102]. Celui-ci construit une première famille de solutions de la forme $\sum_{\nu\in\mathbb{N}} c_\nu\, z^{\lambda+\nu}$, et ramène par « variation de la constante » la recherche des autres solutions à la résolution d'équations d'ordre plus petit. Une partie des résultats se trouvait en fait déjà dans des travaux de Riemann non publiés de son vivant.

Quelques années plus tard, Frobenius [101] propose une preuve plus simple du résultat de Fuchs. La méthode de Frobenius[1] permet de déterminer simplement, par récurrence, les coefficients $u_{k,n}$ de chacune des séries u_k de l'équation (4.1). Elle est devenue la manière « standard » de calculer les solutions (4.1), tant à la main [95, 132, 187] que sur ordinateur [253, 152, 230, 55]. Nous y reviendrons en §4.3.

Parallèlement, au cours des années 1870, Hamburger et d'autres s'intéressent au cas des systèmes différentiels. Ils relient les résultats de Fuchs à ceux de Jordan sur la forme canonique des matrices qui porte son nom, aboutissant peu ou prou à la construction d'une base de solutions esquissée en §4.2, elle aussi devenue classique.

Enfin, Heffter détaille dans son livre de 1894 [126] une alternative à la méthode de Frobenius, variante par la suite modernisée et simplifiée par Poole [203]. Alors que l'on peut soutenir que la méthode de Heffter-Poole est mieux appropriée que celle de Frobenius au calcul formel sur ordinateur, elle semble avoir été négligée jusqu'à récemment. Abramov *et al.* [6] mentionnent explicitement la méthode de Heffter et en identifient plusieurs occurrences en calcul formel ces dernières années. Van der Hoeven [236, §3.2-3.3] décrit une construction voisine de celle de Poole, apparemment retrouvée à partir de la méthode de Frobenius.

Côté implémentations, la plupart des logiciels capables de déterminer les solutions formelles d'équations différentielles linéaires à coefficients analytiques au voisinage de points singuliers réguliers utilisent la méthode de Frobenius. Exception notable, la commande `DEtools[formal_sol]` de Maple [167, 246] fait appel à une méthode due à van Hoeij [247, §8.1]. Celle-ci n'est pas (à ma connaissance) apparentée à la méthode de Frobenius, au sens où elle ne fournit pas de relation de récurrence sur les coefficients des solutions. Elle dépasse pour cette raison le cadre de notre étude.

Morphismes d'anneaux d'opérateurs. Les méthodes de Frobenius et de Heffter-Poole généralisent donc le calcul récursif des solutions séries en un point singulier régulier. Présenté de la façon la plus directe, celui-ci consiste à injecter dans l'équation une série à coefficients indéterminés, extraire un coefficient, puis faire apparaître une récurrence en simplifiant le résultat. Mais il est commode, pour l'exprimer de manière algorithmique et raisonner dessus, d'y penser plutôt comme l'application d'un

1. Le nom « méthode de Frobenius » désigne parfois plus généralement l'injection d'une série à coefficients indéterminés dans une équation différentielle afin de déterminer de proche en proche le développement d'une solution (proposition 3.7). Par ailleurs, beaucoup de descriptions de la méthode dans le cas de solutions de la forme (4.1) en un point singulier régulier se limitent à l'ordre deux. C'est spécifiquement la méthode de détermination des développements *aux singularités* et pour les équations *d'ordre supérieur* qui nous occupe ici.

morphisme entre anneaux de polynômes ou séries non commutatifs, défini par

$$z \mapsto S_n^{-1}, \qquad \theta \mapsto n, \tag{4.3}$$

(cf. §3.2.2) et qui fait passer directement d'un opérateur différentiel $L(z, \theta)$ à l'opérateur de récurrence $L(S_n^{-1}, n)$ associé.

On se propose ici d'adapter ce « point de vue des anneaux d'opérateurs » au cas singulier, en reformulant dans ce langage la méthode de Poole. L'arrière-pensée est bien sûr son utilisation en calcul formel. Nous verrons qu'elle apparaît alors comme une généralisation directe — un « développement perturbatif » — des règles de réécriture (4.3). On aboutit à des expressions des résultats aussi compactes et maniables, sinon un peu plus, que dans la version de Poole. Notre reformulation n'est pas entièrement gratuite : ce formalisme nous sera utile par la suite pour calculer des bornes sur la précision offerte par les développements en série correspondants (§6.4) et pour gagner un peu sur la complexité de l'évaluation à grande précision au voisinage des points singuliers réguliers (§8.6).

Plan du chapitre. La présentation est volontairement redondante. Dans la section 4.2, je commence par rappeler quelques résultats de la théorie analytique classique des points singuliers dans un cadre plus général que nécessaire (systèmes, points singuliers irréguliers), afin de donner une idée du contexte des développements qui suivent. La suite du chapitre est consacrée à différentes méthodes effectives de calcul des solutions formelles dans des cas plus particuliers : méthode de Frobenius (§4.3), méthode de Poole (§4.4), puis enfin méthode originale de Heffter (§4.5). Ces méthodes diffèrent algorithmiquement, mais aussi par la base de solutions qu'elles construisent, et il s'agit de les comparer suivant ces deux aspects. Seule la méthode de Poole (reformulée) est utilisée dans la suite du mémoire.

4.2 Solutions au voisinage d'un point singulier

Cette section collecte quelques éléments de la théorie *locale* des systèmes différentiels linéaires du premier ordre au voisinage d'une singularité isolée. Par la réduction standard des équations d'ordre r aux systèmes du premier ordre, on en déduit des résultats pour ces dernières. Ces rappels sont là essentiellement pour clarifier le contexte des développement qui vont suivre. On pourra consulter comme références générales les livres de Coddington et Levinson [67, Chap. 4], Henrici [127, Chap. 9] ou encore Wasow [252, Chap. I, II, V].

4.2.1 Étude analytique locale

Le premier phénomène qui distingue points ordinaires et points singuliers est que les solutions ne sont plus — pour employer une terminologie un peu datée — « uniformes » mais « multivaluées » quand la variable « fait le tour » d'un point singulier.

Exemple 4.1. L'équation $z\,y'(z) = 1$ admet pour solutions les $c \log z$, $c \in \mathbb{C}$. Alors que les coefficients de l'équation, écrite sous forme unitaire, n'ont qu'un pôle à l'origine, la solution n'est pas méromorphe, ni même analytique sur un voisinage épointé de celle-ci. Son prolongement analytique le long d'un chemin qui s'enroule autour de 0 aboutit à une autre détermination, par exemple $c\,(\log z + 2\,i\,\pi)$ après un tour dans le sens direct.

De même que les solutions en un point ordinaire s'écrivent naturellement comme des combinaisons linéaires des éléments d'une base canonique, avec des coefficients donnés par les conditions initiales, on aimerait pouvoir considérer la constante arbitraire c comme une « condition initiale » en zéro. ♢

Du point de vue de la ramification, ce comportement sur l'exemple le plus simple qui soit résume la situation générale. En effet, une solution au voisinage d'un point

singulier z_0 d'une équation différentielle linéaire à coefficients analytiques s'écrit localement comme combinaison linéaire à coefficients analytiques de « fonctions multivaluées » simples. Contrairement à ce qu'il se passe dans l'exemple de l'équation d'Euler, les coefficients de la combinaison linéaire présentent en général une singularité en z_0, et ne sont analytiques que sur un voisinage épointé de z_0.

Considérons un système différentiel à coefficients analytiques, pris au voisinage d'une singularité isolée, placée à l'origine, de sa matrice de coefficients. On ne fait pas à ce stade d'hypothèse sur la nature de la singularité, qui peut être une singularité effaçable, un pôle ou une singularité essentielle. Soit $D = \mathcal{D}(0,\rho)$ le disque de rayon ρ centré en 0, et posons $D^\bullet = D \setminus \{0\}$.

THÉORÈME 4.2. Soit $A\colon \mathbb{C} \to \mathbb{C}^{r\times r}$ une fonction à valeurs matricielles analytique sur le disque épointé $D^\bullet$. Le système différentiel linéaire homogène

$$Y'(z) = A(z)\,Y(z)$$

admet une matrice fondamentale[2] de la forme

$$Y(z) = z^K\,S(z) = e^{K\log z}\,S(z) \tag{4.4}$$

où $K \in \mathbb{C}^{r\times r}$ est une matrice de constantes et $S\colon D^\bullet \to \mathbb{C}^{r\times r}$ une fonction analytique. La matrice Y est analytique sur le disque fendu $D \setminus \mathbb{R}_-$. ♢

DÉMONSTRATION (ESQUISSE). On suit le prolongement analytique d'une matrice fondamentale le long d'un chemin faisant le tour de la singularité, et l'on construit K à partir de la matrice de monodromie locale obtenue. Voir par exemple Wasow [252, Chap. I] ou Henrici [127, §9.4]. □

Le choix de fendre D le long des réels négatifs est bien sûr arbitraire. On a des résultats analogues sur tout ouvert simplement connexe de $D^\bullet$ et pour toute détermination du logarithme complexe ; ou sur le revêtement universel de $D^\bullet$.

Quitte à effectuer un changement de base ($K \mapsto P^{-1}\,K\,P$ et $Y \mapsto PYP^{-1}$, où P est une matrice constante), on peut supposer K en forme de Jordan. Si elle est réduite à un bloc

$$K = \begin{pmatrix} \lambda & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda \end{pmatrix}$$

l'exponentielle s'écrit

$$z^K = z^\lambda \begin{pmatrix} 1 & \log z & \frac{\log^2 z}{2} & \cdots & \frac{\log^{m-1} z}{(m-1)!} \\ & \ddots & \ddots & & \vdots \\ & & \ddots & \ddots & \frac{\log^2 z}{2} \\ & & & \ddots & \log z \\ & & & & 1 \end{pmatrix}.$$

Ainsi, les coefficients de $Y(z)$ sont des combinaisons linéaires à coefficients analytiques de termes de la forme $z^\lambda \log^k z$, où λ est une valeur propre de K de multiplicité strictement supérieure à k.

COROLLAIRE 4.3. Soient $a_0, a_1, \ldots, a_{r-1}$ des fonctions analytiques sur $D^\bullet$. L'équation scalaire

$$y^{(r)}(z) + a_{r-1}(z)\,y^{(r-1)}(z) + \cdots + a_0(z)\,y(z) = 0$$

2. C'est-à-dire une fonction $z \mapsto Y(z)$ à valeurs dans $\mathrm{GL}_r(\mathbb{C})$ qui est solution de l'équation où l'inconnue est interprétée comme une matrice ; ou encore, la matrice d'une base de solutions.

admet une base de solutions de la forme

$$z^\lambda \sum_{k=0}^{m-1} u_k(z) \log^k z = z^\lambda \sum_{k=0}^{m-1} \sum_{n=-\infty}^{\infty} u_{k,n}\, z^n \log^k z$$

où les λ sont des nombres algébriques et les u_k des fonctions analytiques sur le disque épointé $D^\bullet$. ♢

Démonstration. Il suffit d'appliquer le théorème 4.2 au système compagnon de l'équation différentielle. Étant analytiques sur $D^\bullet$, les u_k y sont développables en séries de Laurent bi-infinies convergentes. □

4.2.2 Points singuliers réguliers et points singuliers irréguliers

Même une fois mis de côté le facteur z^K, la matrice $S(z)$ de l'équation (4.4) présente en général une singularité à l'origine. L'origine est dit point singulier régulier ou point singulier irrégulier du système suivant la nature de cette singularité. Plus précisément, on introduit la définition suivante.

Définition 4.4. Dans la situation du théorème 4.2, on dit que l'origine est

(a) un *point singulier régulier* du système $Y' = A\,Y$, ou de l'opérateur $\partial - A$, si l'on peut choisir K et S de sorte que S n'ait qu'un pôle à l'origine ;

(b) un *point singulier irrégulier* sinon, c'est-à-dire si pour tout choix de (K, S), la matrice S présente une singularité essentielle à l'origine.

La définition s'étend à tout autre point du plan complexe par translation, et au point à l'infini par le changement de variable $z \mapsto 1/z$.

Un point est respectivement un point singulier régulier ou singulier irrégulier d'une équation scalaire (ou d'un opérateur différentiel) à coefficients analytiques sur $D^\bullet$ suivant que c'est un point singulier régulier ou singulier irrégulier de son système compagnon. ♢

Remarquons que l'on peut remplacer « n'ait qu'un pôle » par « soit analytique » dans la définition 4.4(a) quitte à ajouter à K un multiple entier de la matrice identité. Les singularités apparentes sont des points singuliers réguliers. Assez naturellement, nous regrouperons les points ordinaires — où le système a une solution analytique — et les points singuliers réguliers sous le nom de *points réguliers*.

La caractérisation suivante représente une définition alternative intéressante (voir Henrici [127, Theorem 9.4c]).

Proposition 4.5. L'origine est un point régulier du système $Y'(z) = A\,(z)Y(z)$ si et seulement si pour tout secteur angulaire ouvert

$$\Delta = \{r\, e^{i\theta} : (r > 0) \wedge (\theta_{\min} < \theta < \theta_{\max})\}$$

de sommet à l'origine et d'ouverture $\theta_{\max} - \theta_{\min} < 2\,\pi$, le système admet une matrice fondamentale Y analytique sur $D \cap \Delta$ de croissance « polynomiale »

$$Y(z) = O\big((1/z)^{O(1)}\big)$$

quand $z \to 0$ dans Δ.

De même, l'origine est un point singulier régulier d'une équation différentielle linéaire à coefficients analytiques sur $D^\bullet$ si et seulement si toute solution y de l'équation satisfait $y(z) = O\big((1/z)^{O(1)}\big)$ quand $z \to 0$ en restant confiné à un secteur angulaire de sommet à l'origine. □

La restriction de z à un secteur angulaire est là essentiellement pour éviter la situation où z suivrait une courbe s'enroulant « trop rapidement » autour de l'origine.

4.2.3 Étude formelle

Le sujet principal de ce chapitre n'est pas la théorie analytique mais le calcul effectif de la matrice K ainsi que des coefficients du développement en série de $S(z)$. On recherche des couples (K, S) satisfaisant *formellement* l'équation. Dans les situations favorables qui nous intéressent ici, des théorèmes généraux garantissent que toute solution formelle est convergente, et donne donc naissance à une solution analytique.

Systèmes. Avant de nous restreindre aux équations scalaires, résumons rapidement la situation des systèmes du premier ordre. On considère toujours le système $Y' = A\,Y$ avec A analytique sur $D^\bullet = \{z \in \mathbb{C} : 0 < |z| < \rho\}$. On dispose d'une condition *suffisante* simple pour qu'un point singulier donné soit régulier [252, Chap. II].

PROPOSITION 4.6. Un pôle *d'ordre un* de la matrice $A(z)$ est un point régulier. La réciproque est fausse. ♢

Un point où la matrice $A(z)$ a un pôle d'ordre un est appelé singularité de première espèce du système, un point où elle a un pôle d'ordre supérieur, singularité de seconde espèce. Ces notions ont un sens même pour un système à coefficients séries ou séries de Laurent formelles. Dans le cas d'une singularité de première espèce, il est relativement facile de calculer effectivement la matrice K de (4.4) ainsi que le développement en série de $S(z)$ (voir Henrici [127, §9.5] ou Wasow [252, §17.1]).

PROPOSITION 4.7. Soit $A \in \mathbb{C}[[z]]$ une matrice de séries formelles. Le système

$$z\,Y'(z) = A(z)\,Y(z)$$

admet une matrice fondamentale de la forme $Y(z) = z^K\,S(z)$ où $K \in \mathbb{C}^{r\times r}$ est une matrice de constantes et $S \in \mathbb{C}[[z]]^{r\times r}$ une matrice de séries formelles. Les coefficients de K et de S se calculent algébriquement à partir de ceux de A. De plus, si la matrice A des coefficients est analytique sur le disque D, alors S l'est aussi. □

La proposition 4.7 ne résout pas complètement la situation des points singuliers réguliers. Cependant, on peut montrer que si 0 est un point singulier régulier d'un système à coefficients analytiques, il existe une matrice $T(z)$ analytique sur $D^\bullet$, avec au plus un pôle en 0, telle que le changement de variable (« transformation de jauge ») $Y(z) = T(z)\,\hat{Y}(z)$ transforme $Y' = A\,Y$ en un système dont l'origine est singularité de première espèce. Il est possible de préciser ce résultat et de calculer effectivement un tel changement de variable [182]. Tous ces développements dépassent le cadre de notre étude. Je renvoie à l'introduction de Barkatou et Pflügel [13] pour une description plus complète des résultats et plus de références.

Équations scalaires. La situation est nettement plus simple pour les équations scalaires : la nature des points singuliers se lit directement sur l'équation. Singularités de première espèce et points singuliers réguliers se confondent.

THÉORÈME 4.8. (Critère de Fuchs)

(a) [Forme locale.] Soient $a_0, \ldots, a_r \in \mathbb{C}[[z]]$ des séries dont au moins une possède un terme constant non nul. L'origine est un point régulier de

$$L = a_r(z)\,\theta^r + a_{r-1}(z)\,\theta^{r-1} + \cdots + a_0(z) \tag{4.5}$$

si et seulement si $a_r(0) \neq 0$.

(b) [Forme globale.] Soient $a_0, \ldots, a_{r-1}$ des fonctions analytiques sur un ouvert U, sauf en des singularités isolées. Un point $z_0 \in U$ est un point régulier de

$$L = \partial^r + a_{r-1}(z)\,\partial^{r-1} + \cdots + a_0(z),$$

si et seulement si pour tout k, le point z_0 est un pôle de a_k d'ordre inférieur ou égal à $r-k$ (et éventuellement nul). ♢

Tout opérateur $L \in \mathbb{C}((z))\langle\partial\rangle$ peut se mettre (de façon unique) sous la forme requise par le critère de Fuchs local en le multipliant par une puissance de z, ce qui ne change ni l'espace de ses solutions, ni donc la nature de sa singularité éventuelle à l'origine. Nous dirons aussi (un peu abusivement) que l'origine est un point singulier régulier d'une équation différentielle à coefficients *séries formelles* si celle-ci satisfait formellement le critère de Fuchs.

Définition 4.9. Dans la situation du théorème 4.8(a), on appelle *polynôme indiciel* en zéro de l'opérateur différentiel (4.5) (ou de l'équation $L\cdot y=0$) le polynôme

$$Q_0(n) = a_r(0)\, n^r + a_{r-1}(0)\, n^{r-1} + \cdots + a_0(n),$$

et *équation indicielle* l'équation $Q_0(n)=0$. On appelle polynôme et équation indiciels d'un opérateur $L\in\mathbb{C}((z))\langle\partial\rangle$ ceux de son unique multiple $z^j L$ de la forme (4.5)[3]. ♢

Le polynôme indiciel n'est autre que le terme de tête (éventuellement translaté) de la récurrence (3.5) du chapitre précédent. Nous reviendrons longuement sur son rôle dans la suite. Notons que multiplier l'opérateur par une fonction analytique, et en particulier le diviser par $a_0(z)$ lorsque $a_0(z)\neq 0$, ne change le polynôme indiciel que d'un facteur constant. Ses racines restent les mêmes.

Le critère de Fuchs a la conséquence immédiate suivante.

Proposition 4.10. Les points réguliers d'une équation différentielle d'ordre r sont exactement ceux où l'équation indicielle est de degré r. □

En un point singulier régulier d'une équation scalaire, le corollaire 4.3 se précise comme suit.

Théorème 4.11. Soient a_0, ..., a_r des fonctions analytiques au voisinage de zéro. Supposons que l'origine est un point singulier régulier de l'équation

$$a_r(z)\, y^{(r)}(z) + a_{r-1}(z)\, y^{r-1}(z) + \cdots + a_0(z)\, y(z) = 0.$$

Alors celle-ci admet r solutions formelles linéairement indépendantes, de la forme

$$y(z) = z^\lambda \sum_{k=0}^{t-1} u_k(z) \log^k z, \tag{4.6}$$

avec $\lambda\in\mathbb{C}$ et $u_k(z)\in\mathbb{C}[[z]]$ et $t\in\mathbb{N}$. En outre, on peut prendre comme valeurs de λ les r racines de l'équation indicielle, comptées avec multiplicités.

Les séries u_k convergent sur le disque centré à l'origine s'étendant jusqu'à la singularité de $a_0/a_r, \ldots, a_{r-1}/a_r$ la plus proche de l'origine. □

À la suite de Coddington et Levinson [67, Chap. 4], nous appellerons *sommes formelles logarithmiques* (ou *séries logarithmiques*) les combinaisons linéaires d'expressions formelles de la forme z^λ ou $\log^k z$ à coefficients séries de Laurent formelles.

Les solutions (4.6) représentent une base de solutions analytiques de l'équation sur tout secteur angulaire ouvert de sommet à l'origine contenu dans ce disque fendu le long d'une demi-droite, ou dans le revêtement universel du disque épointé. De même que les solutions de la forme

$$y(z) = (z-z_0)^j + O\big((z-z_0)^r\big)$$

3. On peut définir le polynôme indiciel dans la situation de la proposition 4.7, comme le polynôme caractéristique de $A(0)$.

en un point ordinaire z_0 constituent une base privilégiée, il sera utile par la suite de définir une base canonique de solutions au voisinage d'un point singulier régulier : voir §4.4.3.

4.3 La méthode de Frobenius

La méthode de Frobenius est, nous l'avons dit, la plus classique des manières de construire effectivement les solutions (4.6) promises par le théorème de Fuchs. Ne serait-ce que pour cette raison, il semble utile de résumer ici son fonctionnement, bien que ce ne soit pas la méthode que nous adopterons dans la suite. Le lecteur pressé peut sans dommage passer directement à la section suivante.

Cette méthode est présentée dans un grand nombre de traités abordant les équations différentielles linéaires dans le domaine complexe, mais souvent uniquement dans le cas particulier de l'ordre deux. Parmi les descriptions qui couvrent l'ordre supérieur, chacune avec ses petites variantes, citons celles de Forsyth [95, §36], Ince [132, Chap. 16], Poole [168, §19] et Murphy [187, §6-2].

À ma connaissance, sa première implémentation sur ordinateur est due à Watanabe [253] à la fin des années 1960. Par la suite, Tournier [230, chap. 2] ainsi que Chudnovsky et Chudnovsky [54] décrivent la méthode générale, et Lafferty [152] un cas particulier, en lien avec des implémentations dans des systèmes de calcul formel majeurs (Reduce, Scratchpad et Maxima respectivement). La présente discussion emprunte à plusieurs de ces travaux [132, 203, 230, 55].

4.3.1 Le cas générique

Considérons une équation différentielle linéaire

$$a_r(z)\, y^{(r)}(z) + a_{r-1}(z)\, y^{r-1}(z) + \cdots + a_0(z)\, y(z) = 0, \qquad a_i \in \mathbb{C}[[z]], \tag{4.7}$$

dont on suppose que l'origine est un point régulier.

Le point de départ de la méthode de Frobenius est la récurrence « non bornée » déjà vue (§3.2.2) associée à l'équation (4.7). Si $(y_n)_{n\in\mathbb{Z}}$ représente la suite, nulle sur les négatifs, des coefficients d'une solution

$$y(z) = \sum_{n=0}^{\infty} y_n\, z^n,$$

on a une relation

$$Q_0(n)\, y_n + Q_1(n)\, y_{n-1} + \cdots + Q_n(n)\, y_0 = 0, \qquad Q_j \in \mathbb{C}[z], \tag{4.8}$$

que l'on peut encore écrire

$$\sum_{j=0}^{\infty} Q_j(n)\, S_n^{-j} \cdot (y_n) = 0.$$

Le polynôme Q_0 est le polynôme indiciel de l'équation différentielle. La collection des équations (4.8) forme un système triangulaire

$$\begin{aligned} &Q_0(0)\, y_0 = 0 \\ &Q_0(1)\, y_1 = -Q_1(1)\, y_0 \\ &Q_0(2)\, y_2 = -Q_1(2)\, y_1 - Q_1(2)\, y_0 \\ &\cdots \end{aligned} \tag{4.9}$$

Si $y_0 \neq 0$, la première équation entraîne $Q_0(0) = 0$. Lorsque de plus $Q_0(n) \neq 0$ pour $n > 0$, les suivantes permettent de déterminer tous les y_n à partir de y_0, fournissant ainsi une solution pour tout choix de y_0. Si, en revanche, $Q_0(n)$ s'annule pour un certain $n > 0$, l'équation (4.8) induit une relation linéaire entre y_0, ..., y_{n-1} qui, combinée au reste du système, implique en général $y_0 = \cdots = y_{n-1} = 0$. Si d'extraordinaire les contraintes obtenues admettent une solution non triviale, y_n peut être choisi arbitrairement.

Plus généralement, avec $\lambda \in \mathbb{C}$ quelconque, les coefficients d'une solution de la forme

$$y(z) = \sum_{n \in \lambda + \mathbb{Z}}^{\infty} y_n \, z^n \in z^\lambda \, \mathbb{C}[[z]], \qquad y_\lambda \neq 0 \tag{4.10}$$

(étendus par $y_n = 0$ si $n \notin \lambda + \mathbb{N}$) doivent satisfaire (4.8) pour $n \in \mathbb{C}$. Le même raisonnement donne la condition $Q_0(\lambda) = 0$ et permet de construire une solution de la forme (4.10) si $Q_0(\lambda+1), Q_0(\lambda+2), \ldots \neq 0$.

Définition 4.12. Dans le contexte d'un calcul par récurrence, nous appellons *indices exceptionnels* les singularités de la récurrence (4.8), c'est-à-dire les racines de l'équation indicielle, et *indices ordinaires* les autres valeurs de n. Un *groupe*[4] d'indices est un ensemble maximal de racines différant deux à deux d'un entier. ♢

Nous venons de montrer que la *valuation* λ d'une solution (4.10), c'est-à-dire l'exposant de z de partie réelle minimale qui apparaît dans son développement, est nécessairement racine de l'équation indicielle. Inversement, pour chaque groupe G de racines, nous avons construit une solution formelle qui a comme valuation l'élément de G de *partie réelle maximale*. Nous avons vu aussi que toutes les solutions formelles sont convergentes. D'après la proposition 4.10, le polynôme Q_0 est de degré r. Dans la situation où il n'y a ni racines multiples ni racines qui diffèrent d'un entier (qui est celle d'un opérateur singulier régulier « générique », mais pas la plus intéressante en pratique), ces remarques suffisent à obtenir une base de solutions de l'équation (4.7).

4.3.2 Indices exceptionnels

Le théorème de Fuchs prédit plus généralement une solution par racine de l'équation indicielle, quand celles-ci sont comptées avec multiplicité. Comment alors prolonger une solution candidate au-delà des indices exceptionnels ? Frobenius a recours à l'astuce suivante[5]. On introduit un paramètre formel λ, et l'on pose

$$\mathcal{Q}(\lambda) = Q_0(\lambda) \, Q_0(\lambda+1) \cdots Q_0(\lambda + \sigma_{\max})$$

où $\sigma_{\max}$ est la plus grande différence entière entre deux racines de Q_0 (ou entre deux racines d'un certain groupe donné). On remplace provisoirement (4.7) par l'équation inhomogène

$$L(z, \theta) \cdot y = \mathcal{Q}(\lambda) \, z^\lambda, \tag{4.11}$$

qui s'y réduit lorsque λ ou l'un de ses décalés $\lambda - \sigma$ avec $\sigma \leqslant \sigma_{\max}$ est racine de Q_0. Si l'on est à la recherche d'une solution série de coefficients supportés par $\lambda + \mathbb{N}$, cela revient à multiplier la solution candidate (4.10) par $\mathcal{Q}(\lambda)/Q_0(\lambda)$ et ignorer la première équation du système (4.9).

L'intérêt de l'équation auxiliaire (4.11) provient de ce qu'elle admet une solution formelle

$$y(\lambda, z) = z^\lambda \sum_{\nu=0}^{\infty} y_{\lambda+\nu}(\lambda) \, z^\nu \in z^\lambda \, \mathbb{C}(\lambda)[[z]] \tag{4.12}$$

à coefficients fractions rationnelles en λ. En outre, le choix de $\mathcal{Q}$ a pour conséquence que les dénominateurs des $y_n(\lambda)$ ne s'annulent jamais aux racines de Q_0. Cela permet de spécialiser λ à l'une de ces valeurs pour obtenir une solution de l'équation initiale, de valuation λ si la spécialisation $y_\lambda(\lambda)$ ne s'annule pas.

Dérivons formellement l'équation auxiliaire par rapport à λ. On a, au membre gauche,

$$\frac{\partial^m}{\partial \lambda^m} (L \cdot y) = L \cdot \left(\frac{\partial^m y}{\partial \lambda^m} \right) = \sum_{k=0}^{m} \binom{m}{k} \log^k z \sum_{\nu=0}^{\infty} \frac{\partial^m}{\partial \lambda^m} \, y_{\lambda+\nu}(\lambda) \, z^{\lambda+\nu},$$

4. Pour reprendre la terminologie de Fuchs, peut-être un peu malheureuse de nos jours.

5. Tournier [230, p. 37] attribue l'idée de la multiplication par $\mathcal{Q}$ à Kronecker, mais j'ai été incapable de retrouver l'origine de cette référence.

tandis que le membre droit

$$\frac{\partial^m}{\partial\lambda^m}\Big(\mathcal{Q}(\lambda)\,z^\lambda\Big)$$

s'annule dès que l'on substitue à λ une racine de Q_0 de multiplicité $m+1$ ou plus. Il vient dans ce cas

$$L\cdot\left(\sum_{k=0}^{m}\binom{m}{k}\log^k z\sum_{\nu=0}^{\infty}\frac{\partial^m}{\partial\lambda^m}\,y_{\lambda+\nu}(\lambda)\,z^{\lambda+\nu}\right)=0. \tag{4.13}$$

Notons $\mu(n)$ la multiplicité de n comme racine de Q_0 (avec $\mu(n)=0$ si $Q_0(n)\neq 0$). Chaque racine λ_0 de Q_0 est racine de $\mathcal{Q}$ avec multiplicité

$$\mu'(\lambda_0)=\mu(\lambda_0)+\mu(\lambda_0+1)+\cdots+\mu(\lambda_0+\sigma_{\max}),$$

ce qui permet d'appliquer la formule (4.13) avec $0\leqslant m<\mu'(\lambda_0)$ et $\lambda=\lambda_0$. Cependant, λ_0 est aussi racine de $y_\lambda(\lambda)$ avec multiplicité $\mu'(\lambda_0)-\mu(\lambda_0)$. Les solutions trouvées en prenant $0\leqslant m<\mu'(\lambda_0)-\mu(\lambda_0)$ ne contiennent donc pas de terme en $z^{\lambda_0}\log^k z$, et sont combinaisons linéaires de celles apparues pour $\lambda_0=\lambda+\sigma$ avec $\sigma>0$. On peut montrer en revanche que les r solutions restantes, c'est-à-dire celles obtenues en choisissant $\mu'(\lambda_0)-\mu(\lambda_0)\leqslant m<\mu'(\lambda_0)$ dans (4.13) puis en évaluant en λ_0, sont linéairement indépendantes. Elles forment donc une base des solutions de l'équation (4.7).

4.3.3 Base de solutions construite par la méthode de Frobenius

Pour chaque racine λ de Q_0 de multiplicité $\mu=\mu(\lambda)$, apparaissent dans la base ainsi déterminée μ éléments de la forme

$$\begin{aligned}
&v_0(z)\\
&v_0(z)\log z+v_1(z)\\
&\vdots\\
&v_0(z)\log^{\mu-1}z+\mu\,v_1(z)\log^{\mu-1}z+\binom{\mu}{2}v_2(z)\log^{\mu-2}z+\cdots+v_{\mu-1}(z)
\end{aligned}$$

où $v_k(z)\in z^\lambda\,\mathbb{C}[[z]]$. La forme exacte de la base dépend de détails arbitraires de l'algorithme. En particulier, modifier légèrement le facteur $\mathcal{Q}(\lambda)$ (en limitant $\sigma_{\max}$ à la différence maximale entre racines d'un groupe donné, par exemple, ou au contraire en y ajoutant des facteurs superflus) conduit à des solutions différentes après dérivation. Je ne connais pas de bonne caractérisation intrinsèque du résultat de l'algorithme de Frobenius.

EXEMPLE 4.13. Considérons l'équation différentielle

$$z^3\,y^{(4)}(z)-2\,z\,y''(z)+4\,y'(z)-y(z)=0.$$

Son polynôme indiciel vaut $n^2\,(n-3)^2$, et comporte donc à la fois des racines multiples et décalées d'un entier.

Une base « de Frobenius » (celle donnée par les formules d'Ince [132], à ceci près que l'on a normalisé les solutions de sorte que la série en facteur de la puissance maximale de $\ln z$ soit la même dans toutes), dont je ne détaille pas le calcul, est formée des solutions suivantes :

$$\begin{aligned}
F[1]&=\left(z^3+\frac{1}{16}z^4+O(z^5)\right)\\
F[2]&=\left(z^3+\frac{1}{16}z^4+O(z^5)\right)\ln(z)+\left(\frac{77}{30}z^3+\frac{1}{240}z^4+O(z^5)\right)\\
&=\left(\frac{77}{30}+\ln(z)\right)z^3+\left(\frac{1}{240}+\frac{1}{16}\ln(z)\right)z^4\\
F[3]&=\left(z^3+\frac{1}{16}z^4+O(z^5)\right)\ln(z)^2+2\left(\frac{1}{3}z^3-\frac{13}{96}z^4+O(z^5)\right)\ln(z)
\end{aligned}$$

$$+\left(288+72\,z+18\,z^2+\frac{5}{6}\,z^3+\frac{181}{384}\,z^4+O(z^5)\right)$$
$$=288+72\,z+18\,z^2+\left(\frac{5}{6}+\frac{2}{3}\ln(z)+\ln(z)^2\right)z^3+\left(\frac{181}{384}-\frac{13}{48}\ln(z)+\frac{1}{16}\ln(z)^2\right)z^4+\tilde{O}(z^5)$$
$$F[4]=\left(z^3+\frac{1}{16}\,z^4+O(z^5)\right)\ln(z)^3+3\left(\frac{1}{3}\,z^3-\frac{13}{96}\,z^2+O(z^5)\right)\ln(z)^2$$
$$+3\left(288+72\,z+18\,z^2+\frac{5}{6}\,z^3+\frac{181}{384}\,z^4+O(z^5)\right)\ln(z)+\left(864+54\,z^2+\frac{10}{9}\,z^3-\frac{583}{288}\,z^4+O(z^5)\right)$$
$$=864+864\ln(z)+216\,z\ln(z)+(54+54\ln(z))\,z^2+\left(\frac{10}{9}+\frac{5}{2}\ln(z)+\ln(z)^2+\ln(z)^3\right)z^3$$
$$+\left(\frac{-583}{288}+\frac{181}{128}\ln(z)-\frac{13}{32}\ln(z)^2+\frac{1}{16}\ln(z)^3\right)z^4+\tilde{O}(z^5).$$

En anticipant un peu sur la suite, nous pouvons la comparer avec la *base canonique* mieux spécifiée définie en §4.4.3, et qui est dans cet exemple :

$$C[1]=\ln(z)+\left(-\frac{1}{4}+\frac{1}{4}\ln(z)\right)z+\frac{1}{16}\ln(z)\,z^2+\left(-\frac{1}{432}\ln(z)^2+\frac{1}{864}\ln(z)^3\right)z^3$$
$$+\left(-\frac{419}{110592}+\frac{281}{110592}\ln(z)-\frac{19}{27648}\ln(z)^2+\frac{1}{13824}\ln(z)^3\right)z^4+\tilde{O}(z^5)$$
$$C[2]=1+\frac{1}{4}\,z+\frac{1}{16}\,z^2+\frac{1}{288}\ln(z)^2\,z^3+\left(\frac{67}{36864}-\frac{5}{4608}\ln(z)+\frac{1}{4608}\ln(z)^2\right)z^4+\tilde{O}(z^5)$$
$$C[3]=z^3\ln(z)+\left(-\frac{5}{32}+\frac{1}{16}\ln(z)\right)z^4+\tilde{O}(z^5)$$
$$C[4]=z^3+\frac{1}{16}\,z^4+O\left(z^5\right).$$

Un et un seul des monômes privilégiés

$$\ln z,\quad 1,\quad z^3\ln z,\quad z^3$$

est non nul dans chacune des sommes logarithmiques $C[1]$, ..., $C[4]$. On lit grâce à cette observation le système triangulaire de relations

$$F[1]=C[4],\quad F[2]=C[3]+\frac{77}{30}\,C[4],\quad F[3]=288\,C[2]+\frac{2}{3}\,C[3]+\frac{5}{6}\,C[4],$$
$$F[4]=864\,C[1]+864\,C[2]+\frac{5}{2}\,C[3]+\frac{10}{9}\,C[4].\qquad\diamondsuit$$

4.3.4 Récurrences sur les coefficients

Reprenons la relation de récurrence satisfaite par les coefficients de la solution de l'équation auxiliaire (4.11), soit

$$Q_0(\lambda+\nu)\,y_{\lambda+\nu}(\lambda)+Q_1(\lambda+\nu)\,y_{\lambda+\nu-1}(\lambda)+\cdots+Q_\nu(\lambda+\nu)\,y_\lambda(\lambda)=0\qquad(4.14)$$

pour $\nu>0$, et dérivons m fois par rapport à λ. (Il ne faut pas confondre ici la dérivation par rapport à λ de la série $y(\lambda,z)$, qui fait apparaître des termes logarithmiques, et celle des fractions rationnelles $y_{\lambda+\nu}(\lambda)$.) Il vient

$$\sum_{k=0}^{m}\binom{m}{k}\left(Q_0^{(k)}(\lambda+\nu)\,\frac{\partial^{m-k}}{\partial\lambda^{m-k}}\,y_{\lambda+\nu}(\lambda)+\cdots+Q_\nu^{(k)}(\lambda+\nu)\,\frac{\partial^{m-k}}{\partial\lambda^{m-k}}\,y_\lambda(\lambda)\right)=0,$$

égalité qui détermine

$$\frac{\partial^m y_{\lambda+\nu}}{\partial\lambda^m}(\lambda)$$

en fonction des coefficients « précédents »

$$\frac{\partial^k y_{\lambda+\nu'}}{\partial\lambda^k}(\lambda)\qquad\text{avec } \nu'<\nu \text{ ou } \nu'=\nu \text{ et } k<m.$$

Cela vaut même après spécialisation de λ, pourvu seulement que $Q_0(\lambda) \neq 0$.

Dans le cas où l'équation différentielle (4.7) est à coefficients polynomiaux et la récurrence (4.14) d'ordre s, on aboutit au système de récurrences inhomogènes (4.2) annoncé en introduction. Celui-ci fournit les coefficients non seulement de v_0 mais aussi des séries v_k apparues au fil des dérivations par rapport à λ *sans passer par le calcul explicite* de $y(\lambda, z)$ avec λ formel.

Cette observation, particulièrement utile dans les applications en calcul formel qui vont suivre, est explicitée par Chudnovsky et Chudnovsky [55, §6]. Curieusement, elle semble avoir été négligée auparavant : elle ne fait pas partie des présentations habituelles de la méthode de Frobenius, et les implémentations antérieures de Watanabe [253], Lafferty [152] ou Tournier [230] ne l'utilisent pas. Cependant, la méthode de Heffter-Poole, connue — quoique peu répandue — depuis la fin du dix-neuvième siècle, aboutit pratiquement au même système de récurrences de manière plus directe et explicite. Nous passons maintenant à l'étude de cette dernière, qui sera aussi l'occasion d'un examen plus détaillé du système en question.

4.4 La méthode de Poole et la base canonique

4.4.1 Introduction

La méthode de Poole est celle qui aura notre préférence. Nous l'étudions donc de manière plus détaillée que celle de Frobenius.

Soit $\mathbb{K} \subset \mathbb{C}$ un corps. On considère une équation différentielle linéaire à coefficients séries, eux-mêmes à coefficients dans $\mathbb{K}$, que l'on écrit sous la forme

$$L(z, \theta) \cdot y = \Big(a_r(z)\, \theta^r + a_{r-1}(z)\, \theta^{r-1} + \cdots + a_0(z) \Big) \cdot y = 0, \qquad a_k \in \mathbb{K}[[z]], \tag{4.15}$$

où L est une série formelle en deux indéterminées non commutatives. On suppose que $a_r(0) \neq 0$, c'est-à-dire d'après le critère de Fuchs que l'origine est un point régulier. Nous avons vu qu'il existe alors une base de solutions formelles de (4.15) chacune de la forme

$$y(z) = z^\lambda \sum_{k=0}^{t-1} u_k(z) \log^k z, \tag{4.16}$$

où $\lambda \in \mathbb{C}$, les $u_k \in \mathbb{C}[[z]]$, et t est un entier ; et que si les séries a_k sont convergentes, toutes les solutions formelles convergent aussi.

J'appelle méthode de Heffter-Poole, ou simplement de Poole, la « version modifiée », exposée par Poole dans son livre de 1936 [203, §16] d'une alternative à la méthode de Frobenius elle-même proposée par Heffter et décrite en détail dans son ouvrage de 1894 [126, Kap. VIII]. La version présentée ici est une reformulation de celle de Poole, qui vise à la fondre dans le point de vue général des morphismes d'algèbres d'opérateurs utilisé ailleurs dans ce mémoire pour passer d'une équation différentielle à une récurrence sur les développements de Taylor (§3.2.2, §3.3.2) ou de Tchebycheff (§9.2.2) de ses solutions.

La méthode de Heffter-Poole construit une base de solutions formelles de (4.15) en passant par le système de récurrences (4.2). En comparaison de celle qui résulte de la méthode de Frobenius, cette base est facile à caractériser et commode dans les applications. Nous l'appellerons *base canonique*, suivant en cela le consensus de travaux récents en calcul formel [236, 172].

4.4.2 Formulation compacte

Sur la base de la forme générale (4.16), on recherche des solutions sous la forme

$$y(z) = \sum_{n \in \lambda + \mathbb{Z}} \sum_{k \geqslant 0}^{\text{(finie)}} y_{n,k} \frac{\log^k z}{k!}\, z^n. \tag{4.17}$$

On appelle suite des coefficients de y, et l'on note $\boldsymbol{y}$, la suite double

$$\boldsymbol{y} = (y_{n,k})_{\substack{n\in\lambda+\mathbb{Z}\\ k\in\mathbb{N}}};$$

et l'on pose $y_n = (y_{n,k})_{k\in\mathbb{N}}$. On introduit deux opérateurs de décalage S_n et S_k, qui agissent sur les suites doubles par

$$S_n\cdot\boldsymbol{y} = (y_{n+1,k})_{n,k}, \qquad S_k\cdot\boldsymbol{y} = (y_{n,k+1})_{n,k}.$$

Notre reformulation de la méthode de Poole repose sur l'observation — quasiment tautologique — suivante.

Proposition 4.14. Une somme logarithmique $y(z)\in z^\lambda\,\mathbb{C}[[z]][\log z]$ est solution formelle de l'équation différentielle $L(z,\,\theta)\cdot y = 0$ si et seulement si sa suite de coefficients $\boldsymbol{y}$ satisfait

$$L(S_n^{-1}, n+S_k)\cdot\boldsymbol{y} = 0. \tag{4.18}$$

On a $L(S_n^{-1}, n+S_k)\in\mathbb{K}[n+S_k][[S_n^{-1}]]$. ♢

Démonstration. Il suffit d'écrire l'action des opérateurs z et θ sur les sommes logarithmiques. Comme

$$\theta\cdot z^n = n\,z^n \qquad \text{et} \qquad \theta\cdot\frac{\log^k z}{k!} = \frac{\log^{k-1} z}{(k-1)!} \quad (k\geqslant 1),$$

on a

$$\begin{aligned}
z\cdot y(z) &= \sum_{n\in\lambda+\mathbb{Z}}\sum_{k\geqslant 0} y_{n,k}\,\frac{\log^k z}{k!}\,z^{n+1} = \sum_{n\in\lambda+\mathbb{Z}}\sum_{k\geqslant 0} y_{n-1,k}\,\frac{\log^k z}{k!}\,z^n\\
\theta\cdot y(z) &= \sum_{n\in\lambda+\mathbb{Z}}\left(\sum_{k\geqslant 0} n\,y_{n,k}\,\frac{\log^k z}{k!}\,z^n + \sum_{k\geqslant 1} y_{n,k}\,\frac{\log^{k-1} z}{(k-1)!}\,z^n\right)\\
&= \sum_{n\in\lambda+\mathbb{Z}}\sum_{k\geqslant 0}(n\,y_{n,k}+y_{n,k+1})\,\frac{\log^k z}{k!}\,z^n.
\end{aligned}$$

La suite des coefficients de $L(z,\theta)\cdot y$ est donc $L(S_n^{-1}, n+S_k)\cdot\boldsymbol{y}$. □

Exemple 4.15. Considérons l'équation $z^3\,y''' + z^2\,y'' - z\,y' - z\,y = 0$, qui s'écrit encore

$$L(z,\theta)\cdot y = (\theta^3 - 2\,\theta^2 - z)\cdot y = 0.$$

On a $L(S_n^{-1}, n+S_k) = Q_0(n+S_k) - S_n^{-1}$, où

$$Q_0(n+S_k) = n^2\,(n-2) + n\,(3\,n+4)\,S_k + (3\,n-2)\,S_k^2 + S_k^3.$$

Cet exemple est emprunté à van der Hoeven [236, §3.4]. Nous le filerons dans les pages qui viennent jusqu'à déterminer les développements d'une base de solutions. ♢

Nous baptisons la relation (4.18) *récurrence implicite* associée à l'équation différentielle. Il s'agit en effet d'une équation aux différences vérifiée par la suite des coefficients d'une solution, qui ne suffit toutefois pas en l'état à calculer commodément ceux-ci à partir de valeurs initiales.

Écrivons la division euclidienne à droite de $L(z,\theta)$ par z :

$$L(z,\theta) = Q_0(\theta) + R(z,\theta)\,z.$$

Le reste $Q_0(\theta)\in\mathbb{K}[\theta]$ est simplement le polynôme indiciel de $L(z,\theta)$ à l'origine. Pour tout n, on note $\mu(n)$ la multiplicité de n comme racine de Q_0. Le quotient $R(z,\theta)$ est un polynôme ou une série formelle en z suivant la nature de Q.

Soit $y(z)$ une solution de $L(z,\theta)\cdot y = 0$ de la forme (4.17). La proposition suivante, due à Poole dans un langage différent, fournit une *récurrence explicite* sur les $y_{n,k}$.

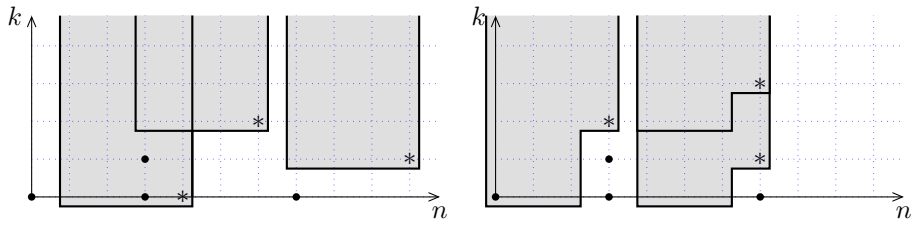


Figure 4.1. Allure des relations déduites de (4.19), en un indice ordinaire (schéma de gauche) ou exceptionnel (schéma de droite). Chaque boîte grisée signale une dépendance linéaire entre les éléments de la suite $(y_{n,k})$ qu'elle englobe. Le coefficient marqué d'une étoile est non nul, de sorte que la valeur de $y_{n,k}$ correspondante est déterminée par le reste de la boîte. Inversement, les $y_{n,k}$ marqués d'un point noir, donnés par l'équation indicielle, peuvent être choisis arbitrairement. L'exemple représenté correspond à une équation différentielle à coefficients polynomiaux, de polynôme indiciel $Q_0(n) \propto n\,(n-3)^2\,(n-7)$, associée à une récurrence d'ordre 3.

PROPOSITION 4.16. [203, §16] La suite $\boldsymbol{y} = (y_{n,k})_{n,k}$ est donnée par la relation de récurrence

$$S_k^{\mu(n)} \cdot \boldsymbol{y} = -T_n(S_k)\, R(S_n^{-1}, n + S_k)\, S_n^{-1} \cdot \boldsymbol{y} \tag{4.19}$$

où, pour tout $n \in \mathbb{C}$ fixé, $T_n(X) \in \mathbb{K}(n)[X]$ désigne le polynôme de degré au plus $(t-1)$ inverse de $X^{-\mu(n)}\, Q_0(n+X)$ modulo X^t. ♢

DÉMONSTRATION. D'après la proposition 4.14, la suite $\boldsymbol{y}$ vérifie

$$Q_0(n + S_k) \cdot \boldsymbol{y} = -R(S_n^{-1}, n + S_k)\, S_n^{-1} \cdot \boldsymbol{y}.$$

Pour $n \in \mathbb{C}$ fixé, l'expression $X^{-\mu(n)}\, Q_0(n+X)$ est un polynôme en X qui ne s'annule pas en zéro, donc inversible dans $\mathbb{K}(n)[[X]]/\langle X^t \rangle$. On obtient ainsi un polynôme T_n, de degré $\deg T_n < t$, tel que

$$T_n(X)\, Q_0(n+X) = X^{\mu(n)} + O(X^t).$$

Or, d'après (4.16), la solution formelle $y(z)$ est de degré en $\log z$ borné par $t-1$, c'est-à-dire que l'on a

$$S_k^t \cdot \boldsymbol{y} = 0$$

et par conséquent

$$S_k^{\mu(n)} \cdot \boldsymbol{y} = T_n(S_k)\, Q_0(n + S_k) \cdot \boldsymbol{y} = -T_n(S_k)\, R(S_n^{-1}, n + S_k) \cdot \boldsymbol{y},$$

le résultat annoncé. □

À multiplicité de n dans Q_0 fixée, $T_n(S_k)$ est un polynôme en S_k à coefficients fractions rationnelles en n sans pôle aux valeurs de n correspondantes. Il peut être commode de l'exprimer explicitement comme

$$T_n(S_k) = \sum_{i=0}^{t-1} \frac{1}{i!} \frac{\partial^i}{\partial X^i} \frac{X^{\mu(n)}}{Q_0(X)} \bigg|_{X=n} S_k^i.$$

En particulier, la valeur de $T_n(S_k)$ pour $\mu(n) = 0$ est facile à calculer en fonction de n. L'injecter dans (4.19) donne une formule de récurrence à coefficients dans $\mathbb{K}$ valable aux indices n ordinaires. La récurrence change de forme aux indices exceptionnels.

En introduisant la notation

$$\boldsymbol{y}_{p,q} = S_n^p\, S_k^q\, \boldsymbol{y}$$

la récurrence explicite (4.19) s'écrit encore sous la forme, peut-être plus intuitive,

$$\boldsymbol{y}_{n,k+\mu(n)} = -T_n(S_k)\, R(S_n^{-1}, n + S_k) \cdot \boldsymbol{y}_{n-1,k}. \tag{4.20}$$

Elle concentre à peu près tout ce qu'il y a à savoir pour décrire efficacement les développements des solutions de l'équation.

On obtient tout d'abord une condition nécessaire pour que puisse exister une solution avec $y_{n^*,k^*} \neq 0$ et $y_n = 0$ pour $n^* - n > 0$: il faut que n^* soit racine de Q_0 avec multiplicité au moins $k^* + 1$, sans quoi la relation de récurrence (4.20) prise en $n = n^*$ et $k = k^* - \mu(n^*) \geqslant 0$ entraîne immédiatement que toute la suite $(y_{n^*,k})_{k \geqslant k^*}$ est nulle. Plus généralement, on voit qu'à n fixé, les $y_{n',k}$ avec $n' < n$ déterminent la suite $(y_{n,k})_{k \geqslant \mu(n)}$, tandis que les $y_{n,k}$ pour $k < \mu(n)$ demeurent libres. Ces dépendances sont illustrées en figure 4.1.

L'équation (4.20) est utilisable directement[6] pour calculer les $y_{n,k}$ à partir de valeurs initiales données aux points laissés libres. Pour chaque $n \in \lambda + \mathbb{N}$ avec λ racine de Q_0, on calcule successivement $y_{n,t-1}, y_{n,t-2}, \ldots, y_{n,0}$ en se servant des coefficients de $y_{n-1}, y_{n-2}, \ldots$ et de la nullité de $y_{n,k} = 0$ pour $k \geqslant t$.

Exemple 4.17. Poursuivons le calcul entamé dans l'exemple 4.15. Lorsque n et S_k sont des indéterminées, on a

$$Q_0(n + S_k)^{-1} = \frac{1}{n^2\,(n-2)} \left(1 + \frac{-3\,n+4}{n\,(n-2)}\, S_k + \frac{2\,(3\,n^2 - 8\,n + 6)}{n^2\,(n-2)^2}\, S_k^2 + O(S_k^3) \right)$$

et donc, pour $n \notin \{0, 2\}$, la récurrence explicite s'écrit

$$\boldsymbol{y}_{n,k} = \frac{1}{n^2\,(n-2)} \left(\boldsymbol{y}_{n-1,k} + \frac{-3\,n+4}{n\,(n-2)}\, \boldsymbol{y}_{\substack{n-1,\\ k+1}} + \frac{2\,(3\,n^2 - 8\,n + 6)}{n^2\,(n-2)^2}\, \boldsymbol{y}_{\substack{n-1,\\ k+2}} \right). \tag{4.21}$$

En l'indice exceptionnel $n = 2$, où celle-ci présente un pôle, on a

$$Q_0(2 + S_k) = (4 + 4\,S_k + S_k^2)\, S_k,$$

d'où le développement

$$S_k\, Q_0(2 + S_k)^{-1} = \frac{1}{4} - \frac{1}{4}\, S_k + \frac{3}{16}\, S_k^2 + O(S_k^3)$$

et la récurrence explicite exceptionnelle

$$\boldsymbol{y}_{2,k+1} = \frac{1}{4}\, \boldsymbol{y}_{1,k} - \frac{1}{4}\, \boldsymbol{y}_{1,k+1} + \frac{3}{16}\, \boldsymbol{y}_{1,k+2}. \tag{4.22}$$

Il n'est pas nécessaire de calculer la récurrence explicite exceptionnelle en $n = 0$. En supposant les y_n pour $n < 0$ nuls, on sait à l'avance qu'elle impose $y_{0,k} = 0$ pour $k \geqslant 2$ et qu'elle laisse $y_{0,1}$ et $y_{0,2}$ libres.

Chaque solution y de l'équation différentielle est ainsi déterminée par le triplet de conditions initiales $(y_{0,0}, y_{0,1}, y_{2,0})$. En posant $S_k^{[-1]} \cdot y_n = (0, y_{n,0}, y_{n,1}, \ldots)$ et en notant abusivement $y_n = \sum_k y_{n,k} \frac{1}{k!} \log^k z$ au lieu de $y_n = (y_{n,0}, y_{n,1}, \ldots)$, on a tour à tour

$$\begin{aligned}
y_0 &= y_{0,0} + y_{0,1} \log z, \\
y_1 &= -(1 - S_k + S_k^2) \cdot y_0 && \text{d'après (4.21)} \\
&= -(y_{0,0} + y_{0,1} \log z) + y_{0,1} \\
&= (-y_{0,0} + y_{0,1}) - y_{0,1} \log z, \\
y_2 &= y_{2,0} + S_k^{-1} \cdot y_2 = y_{2,0} + S_k^{[-1]} \left(\frac{1}{4} - \frac{1}{4}\, S_k + \frac{3}{16}\, S_k^2 \right) \cdot y_1 && \text{d'après (4.22)}
\end{aligned}$$

6. Comme nous le verrons en §8.6, vue comme une récurrence en n, elle se prête aussi à l'évaluation par *scindage binaire*. Cela signifie qu'il est assez facile de calculer efficacement un terme $y_n = (y_{n,k})_k$ de la suite des coefficients sans calculer tous les précédents, en multipliant des matrices de séries tronquées $B(n) \in \mathbb{K}(n)[[S_k]]/\langle S_k^t \rangle$ obtenues en traduisant (4.19) en une récurrence matricielle du premier ordre.

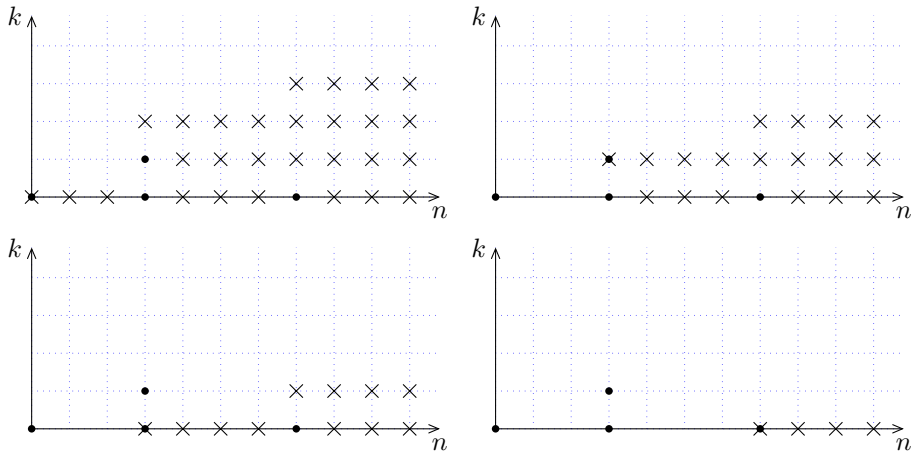


Figure 4.2. Support des solutions de la base canonique, pour une équation différentielle générique de polynôme indiciel $n\,(n-3)^2\,(n-7)$. Les croix représentent les coefficients non nuls (sauf pour certaines équations exceptionnelles) ; les points, les coefficients arbitraires où se placent les conditions initiales.

$$= y_{2,0} + \frac{1}{4}\left((-y_{0,0} + y_{0,1}) \log z - y_{0,1} \frac{\log^2 z}{2} \right) + \frac{1}{4}\, y_{0,1} \log z$$

$$= y_{2,0} + \frac{-y_{0,0} + 2\, y_{0,1}}{4} \log z - \frac{y_{0,1}}{4} \frac{\log^2 z}{2}$$

et ainsi de suite. Le calcul des termes suivants ne fait appel qu'à la relation (4.21). ♢

REMARQUE 4.18. On n'a pas utilisé à ce stade l'hypothèse que l'origine est un point régulier de $L(z,\theta)$, sinon implicitement, en se focalisant sur les suites $(y_{n,k})$ qui s'annulent quand la partie réelle de n est suffisamment négative. Ce qui précède s'applique aux solutions régulières — celles de la forme (4.16) — en un point singulier irrégulier, à ceci près qu'une solution formelle régulière n'est pas forcément convergente. (Ainsi, la série $y(z) = \sum_{n \geqslant 0} n!\, z^n$ est solution formelle de l'équation $z^2\, y'' + (3\, z - 1)\, y' + y = 0$.)

Plus généralement, il résulte du corollaire 4.3 qu'une solution autour d'un point singulier irrégulier est combinaison linéaire de sommes *bi-infinies* convergentes de la forme (4.17). Les coefficients $y_{n,k}$ correspondants doivent eux aussi satisfaire la récurrence (4.19). Toutefois, sans la contrainte que $y_{n,k}$ s'annule quand $\operatorname{Re} n \to -\infty$, cette récurrence a évidemment une infinité de solutions, et l'on a peu de prise sur les conditions de convergence qui distinguent les solutions « intéressantes ». De fait, le traitement habituel des points singuliers irréguliers ne procède pas par cette voie, mais par des changements de variable qui ramènent l'étude *asymptotique* des solutions de l'équation à celle des solutions régulières formelles d'un nombre fini d'équations auxiliaires. (Voir théorème 3.23, et par exemple Wasow [252, Chap. IV] pour plus de détails.) ♢

Il est possible de développer la formule (4.19) pour obtenir un certain nombre de variantes plus classiques, comme nous le verrons en §4.4.4. Mais commençons par définir, toujours à l'aide de cette récurrence, la base canonique des solutions de (4.15).

4.4.3 Base canonique de solutions en un point singulier régulier

COROLLAIRE 4.19. Soit $z_0 \in \mathbb{C}$ un point régulier d'une équation différentielle linéaire à coefficients dans $\mathbb{K}[[z]]$. Pour tout $n \in \mathbb{C}$, soit $\mu(n)$ la multiplicité de n comme racine du polynôme indiciel en z_0 de l'équation. Soit

$$E = \{(n,k) : 0 \leqslant k < \mu(n)\}.$$

Pour tout $e=(n^*,k^*)\in E$, il existe une unique solution

$$y[z_0,e]=\sum_{n\in\lambda+\mathbb{Z}}\sum_{k\in\mathbb{N}} y[z_0,e]_{n,k}\,\frac{\log^k(z-z_0)}{k!}\,(z-z_0)^n$$

de l'équation telle que $y[z_0,e]_{n,k}$ s'annule ultimement quand $\operatorname{Re} n\to-\infty$, et que

$$\begin{cases} y[z_0,(n^*,k^*)]_{n^*,k^*}=1\\ y[z_0,(n^*,k^*)]_{n,k}=0, \quad (n,k)\in E\setminus\{(n^*,k^*)\}.\end{cases} \tag{4.23}$$

La famille $(y[z_0,e])_{e\in E}$ est une base de l'espace des solutions formelles de (4.15). Si les coefficients de (4.15) sont des séries de rayon de convergence au moins ρ et si le point singulier le plus proche est lui-même à distance au moins ρ de z_0, c'est aussi une base des solutions analytiques de (4.15) sur le disque fendu $\mathcal{D}(0,\rho)\setminus\mathbb{R}_-$. ♢

Démonstration. Nous venons de voir comment construire une telle famille à l'aide de la récurrence (4.19). Il est clair d'après les conditions (4.23) que les $y[z_0,e]$ pour $e\in E$ sont linéairement indépendantes. Comme z_0 est supposé point régulier de (4.15), le polynôme indiciel correspondant est de degré r, donc card $E=r$ et ces solutions forment une base. □

Définition 4.20. La famille $(y[z_0,e])_{e\in E}$ définie au corollaire 4.19 est appelée *base canonique* en z_0 des solutions de l'équation. Les coefficients de l'écriture dans cette base d'une solution y quelconque, c'est-à-dire les coefficients de

$$(z-z_0)^n\,\frac{\log^k(z-z_0)}{k!},\qquad (n,k)\in E$$

dans son développement en série logarithmique, sont appelées *conditions initiales généralisées* en z_0 associées à y. ♢

Cette définition généralise celle de la section 3.2.2. La figure 4.2 donne la forme des solutions de base pour l'équation indicielle de la figure 4.1. Finissons aussi notre calcul explicite.

Exemple 4.21. On repart de la solution générale, établie dans les exemples 4.15 et 4.17, de l'équation $z^2y'''+zy''-y'-y=0$. La base canonique est donnée par les choix de conditions initiales

$$(y_{0,0},y_{0,1},y_{2,0})\in\{(1,0,0),(0,1,0),(0,0,1)\}.$$

En poussant les calculs jusqu'au terme en z^3 et en injectant ces différentes conditions initiales, il vient

$$\begin{aligned} y_{[0,0]}(z)&=1-z-\left(\frac14\log z\right)z^2+\left(\frac{5}{108}-\frac{1}{36}\log z\right)z^3+\tilde O(z^4)\\ y_{[0,1]}(z)&=\log z+(1-\log z)\,z+\left(\frac12\log z-\frac18\log^2 z\right)z^2\\ &\quad+\left(\frac{-4}{27}+\frac{11}{108}\log z-\frac{1}{72}\log^2 z\right)z^3+\tilde O(z^4)\\ y_{[2,0]}(z)&=z^2+\frac19 z^3+O(z^4).\end{aligned}$$

Notre expression de $y_{[0,1]}$ est en désaccord avec celle de van der Hoeven [236, §3.4]. ♢

Remarque 4.22. Le polynôme indiciel en un point *ordinaire* d'une équation différentielle d'ordre r est un multiple constant de $n^{\uparrow r}=n\,(n-1)\cdots(n-r+1)$, et possède donc des racines différant d'un entier. On sait pourtant d'après le théorème de Cauchy

qu'il n'y a que des solutions séries. Il est instructif de le vérifier directement à partir de la récurrence (4.19).

Considérons donc l'équation

$$\Big(\partial^r + a_{r-1}(z)\,\partial^{r-1} + \cdots + a_0(z)\Big)\cdot y = 0, \qquad a_i(z) = \sum_{j=0}^{\infty} a_{i,j}\, z^j \in \mathbb{C}[[z]].$$

La récurrence associée s'écrit

$$\begin{aligned} S_n^{-r} \sum_{i=0}^{r} a_j(S_n^{-1})\,(n+i)^{\downarrow i}\, S_n^i &= \sum_{i=0}^{r}\sum_{j=0}^{\infty} a_{i,j}\,(n+i-j-r)^{\downarrow i}\, S_n^{-j-r+i} \\ &= \sum_{i=0}^{\infty}\sum_{j=i}^{i+r} a_{i+r-j,i}\,(n-j)^{\downarrow(r-j+i)}\, S_n^{-j} \end{aligned}$$

d'où, lorsque n est racine simple de l'équation indicielle,

$$\boldsymbol{y}_{n,k+1} = -T_n(S_k) \sum_{i=0}^{\infty}\sum_{j=i}^{i+r} a_{i+r-j,i}\,(n-j+S_k)^{\downarrow(r-j+i)}\, \boldsymbol{y}_{n-j,k}.$$

Soit $n \in [\![0, r-1]\!]$, et supposons $\boldsymbol{y}_{n',1} = 0$ pour tout $n' < n$. On a alors

$$\begin{aligned} \boldsymbol{y}_{n,1} &= -T_n(0) \sum_{i=0}^{\infty}\sum_{j=i}^{i+r} a_{i+r-j,i}\,(n-j)^{\downarrow(r-j+i)}\, \boldsymbol{y}_{n-j,0} \\ &= -T_n(0) \sum_{i=0}^{\infty}\sum_{j=i}^{i+r} a_{i+r-j,i}\,(n-r)^{\downarrow i}\, \underbrace{(n-j)^{\downarrow(r-j)}}_{\text{nul pour } j\leqslant n<r}\, \overbrace{\boldsymbol{y}_{n-j,0}}^{\text{nul pour } j>n} \\ &= 0. \end{aligned}$$

On conclut par récurrence sur n qu'il apparaît jamais de logarithmes. ♢

4.4.4 Jeux d'écriture

Nous terminons l'étude de la méthode de Poole en donnant quelques conséquences ou écritures alternatives des récurrences (4.18) et (4.19). En particulier, nous explicitons le système de relations entre les $y_{n,k}$ qu'elles induisent. Cela permet de faire le lien avec la méthode de Frobenius et avec quelques variantes de la méthode de Heffter apparaissant dans la littérature. Le contenu de cette section n'est invoqué que dans celle qui suit immédiatement.

Récurrence développée. Notons L', L'', ..., $L^{(r)}$, ... les dérivées successives par rapport à V de $L(U,V) \in \mathbb{K}[[U]][V]$. En développant $L(S_n^{-1}, n+S_k)$ par rapport à S_k, on réécrit (4.18) sous la forme

$$\sum_{i=0}^{r} \frac{1}{i!}\, L^{(i)}(S_n^{-1}, n)\cdot \boldsymbol{y}_{n,k+i} = 0.$$

Cette expression est la formulation compacte d'un système de récurrence inhomogènes en n satisfait par les suites $(y_{n,k})_{n\in\lambda+\mathbb{Z}}$ pour tout k fixé. Il est triangulaire : compte tenu que les suites doubles $\boldsymbol{y}_{n,k}$ avec $k \geqslant t$ sont nulles, on obtient en effet

$$\left\{\begin{array}{ll} L(S_n^{-1},n)\cdot \boldsymbol{y}_{n,t-1} &= 0 \\ L(S_n^{-1},n)\cdot \boldsymbol{y}_{n,t-2} &= -L'(S_n^{-1},n)\cdot \boldsymbol{y}_{n,t-1} \\ L(S_n^{-1},n)\cdot \boldsymbol{y}_{n,t-3} &= -L'(S_n^{-1},n)\cdot \boldsymbol{y}_{n,t-2} - \dfrac{L''(S_n^{-1},n)}{2}\cdot \boldsymbol{y}_{n,t-1} \\ & \vdots \\ L(S_n^{-1},n)\cdot \boldsymbol{y}_{n,0} &= -L'(S_n^{-1},n)\cdot \boldsymbol{y}_{n,1} - \cdots - \dfrac{L^{(t-1)}(S_n^{-1},n)}{(t-1)!}\cdot \boldsymbol{y}_{n,t-1}. \end{array}\right. \tag{4.24}$$

On développe à nouveau, cette fois en série par rapport à la première variable (en appliquant les relations de commutation convenables), et l'on pose

$$L(S_n^{-1}, n) = Q_0(n) + Q_1(n)\, S_n^{-1} + Q_2(n)\, S_n^{-2} + \cdots.$$

Lorsqu'on extrait le terme d'indice $(0,0)$ dans les égalités (4.24), c'est-à-dire lorsque l'on « remplace $\boldsymbol{y}_{n,k}$ par $y_{n,k}$ », chaque ligne prend l'allure suivante :

$$\begin{aligned}
=-\Big(&\boxed{Q_0(n)\, y_{\substack{n\\k}}} + Q_1(n)\, y_{\substack{n-1\\k}} + Q_2(n)\, y_{\substack{n-2\\k}} + \cdots \\
&Q_0'(n)\, y_{\substack{n\\k+1}} + Q_1'(n)\, y_{\substack{n-1\\k+1}} + Q_2'(n)\, y_{\substack{n-2\\k+1}} + \cdots \\
+&\frac{Q_0''(n)}{2}\, y_{\substack{n\\k+2}} + \frac{Q_1''(n)}{2}\, y_{\substack{n-1\\k+2}} + \frac{Q_2''(n)}{2}\, y_{\substack{n-2\\k+2}} + \cdots \\
&\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
+&\boxed{\frac{Q_0^{(\nu-1)}(n)}{(\nu-1)!}\, y_{\substack{n\\k+\nu-1}}} + \frac{Q_1^{(\nu-1)}(n)}{(\nu-1)!}\, y_{\substack{n-1\\k+\nu-1}} + \frac{Q_2^{(\nu-1)}(n)}{(\nu-1)!}\, y_{\substack{n-2\\k+\nu-1}} + \cdots \\
&\frac{Q_0^{(\nu)}(n)}{\nu!}\, y_{\substack{n\\k+\nu}} + \frac{Q_1^{(\nu)}(n)}{\nu!}\, y_{\substack{n-1\\k+\nu}} + \frac{Q_2^{(\nu)}(n)}{\nu!}\, y_{\substack{n-2\\k+\nu}} + \cdots \\
&\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
+&\frac{Q_0^{(t-k)}(n)}{(t-k)!}\, y_{\substack{n\\t}} + \frac{Q_1^{(t-k)}(n)}{(t-k)!}\, y_{\substack{n-1\\t}} + \frac{Q_2^{(t-k)}(n)}{(t-k)!}\, y_{\substack{n-2\\t}} + \cdots \Big).
\end{aligned} \tag{4.25}$$

Le bloc encadré s'annule quand n est racine de multiplicité ν de l'équation indicielle (voir figure 4.1).

Si l'équation différentielle est à coefficients polynomiaux, chacune des sommes en ligne s'arrête avec le terme en y_{n-s} pour un certain ordre s, les Q_i étant nuls au-delà. Le système complet des récurrences couplées vérifiées par les coefficients $y_{n,k}$ est la conjonction des égalités (4.25) pour $0 \leqslant k < t$. Il s'agit là de la version entièrement explicite des récurrences inhomogènes (4.2) déjà mentionnées en introduction puis en lien avec la méthode de Frobenius.

On retrouve aussi les formules (3.8-3.9) de van der Hoeven [236], et l'on peut examiner le comportement du système aux indices exceptionnels en paraphrasant pratiquement son étude [236, §3.2.3]. Si $Q_0(n) \neq 0$, le système détermine $y_{n,t}, y_{n,t-1}\ldots$, $y_{n,0}$ en fonction des suites précédentes y_{n-1}, $y_{n-2,}\ldots$ En général, lorsque n est une racine de multiplicité ν de Q_0, on obtient ν relations linéaires « exceptionnelles » entre les $y_{n,k}$, suivant la structure

$$\text{une relation entre}\begin{cases} \text{une relation entre}\begin{cases} \text{une relation entre}\{y_{n-1,t}, y_{n-2,t}, \ldots, \\ \qquad\qquad y_{n-1,t-1}, y_{n-2,t-1}, \ldots, \\ \qquad\qquad \vdots \\ \qquad\qquad y_{n-1,t-\mu(n)+1}, y_{n-2,t-\mu(n)+1}, \ldots, \end{cases} \end{cases}$$

après quoi $y_{n,t}$, $y_{n,t-1}$, …, $y_{n,\mu(n)}$ sont successivement déterminés en fonction des termes connus. Dans la construction de la base canonique, on fait en sorte que ces relations soient trivialement satisfaites, tous les coefficients correspondants étant nuls.

Deux systèmes différentiels. D'autres variantes se présentent si l'on écrit les diverses relations précédentes sous forme de systèmes différentiels plutôt que de récurrences. Surchargeons encore un peu la notation y en posant provisoirement

$$y_n(z) = \sum_{k\geqslant 0}^{(\text{finie})} y_{n,k}\, \frac{\log^k z}{k!}.$$

De l'équation $L(z,\,\theta)\,\cdot\,y\,=\,0$, on tire alors (avec des notations que j'espère évidentes) $L(S_n^{-1}, n+\theta)\cdot(y_n(z))_n$, c'est-à-dire

$$\theta^{\mu(n)}\cdot(y_n(z))_n = -T_n(\theta)\,R(S_n^{-1}, n+\theta)\,S_n^{-1}\cdot(y_n(z))_n.$$

Les y_n étant nuls pour $n\to-\infty$, il s'agit d'un système différentiel triangulaire satisfait par les fonctions $y_n(z)\in\mathbb{C}[\log z]$. C'est là l'analogue de notre proposition 4.16 dans l'exposé de Poole [203, Eq. (13-17)]. Il poursuit en éliminant les $y_n(z)$ pour $Q(n)\neq 0$, afin d'obtenir un système différentiel qui caractérise les choix possibles des $y_n(z)$ associés aux indices exeptionnels.

Enfin, comme nous l'avions fait dans la méthode de Frobenius, on peut choisir de regrouper les coefficients par puissance de z en premier puis seulement par puissance de $\log z$. Posons

$$y(z) = \sum_{k\geqslant 0}\frac{1}{k!}\,w_k(z) \quad \text{soit} \qquad w_k(z) = \sum_{n\in\lambda+\mathbb{Z}} y_{n,k}\,z^n. \tag{4.26}$$

On écrit aussi $\boldsymbol{w}_k = (w_{k+k'})_{k'\geqslant 0}$ et $\boldsymbol{w} = \boldsymbol{w}_0$, suites sur lesquelles on convient de faire agir l'opérateur S_k par décalage, les opérateurs z et θ coefficient par coefficient. La récurrence implicite devient cette fois $L(z,\theta+S_k)\cdot\boldsymbol{w}=0$, soit en développant

$$L(z,\theta)\cdot\boldsymbol{w}_k + L'(z,\theta)\cdot\boldsymbol{w}_{k+1} + \cdots + \frac{1}{r!}\,L^{(r)}(z,\theta)\cdot\boldsymbol{w}_{k+r} = 0. \tag{4.27}$$

S'ensuit l'analogue différentiel des récurrences (4.24)

$$\begin{aligned}
L(z,\theta)\cdot w_{t-1} &= 0\\
L(z,\theta)\cdot w_{t-2} &= -L'(z,\theta)\cdot w_{t-1}\\
&\vdots\\
L(z,\theta)\cdot w_0 &= -L'(z,\theta)\cdot w_1 - \cdots - \frac{1}{(t-1)!}\,L^{(t-1)}(z,\theta)\cdot w_{t-1}.
\end{aligned}$$

On retrouve le système écrit par van der Hoeven [236, Eq. (3.4)].

4.5 La méthode originale de Heffter

Concluons par une section d'intérêt essentiellement historique. Son objet est de faire le lien entre la méthode de Heffter proprement dite et la variante de Poole ou la nôtre. Pour cela, je reprends le canevas de la preuve originale, en faisant appel au formalisme introduit plus haut en lieu et place des manipulations explicites de systèmes linéaires. Les numéros en gras font référence aux articles du chapitre VIII du livre de Heffter [126].

On part à nouveau de l'équation $L(z,\theta)\cdot y=0$. La méthode de Heffter s'applique aux racines de l'équation indicielle prises groupe par groupe. Limitons-nous pour simplifier au groupe formé des racines entières ($\lambda=0$). On pose $F=\mathbb{C}[[z]][\log z]$. On définit w_k, $\boldsymbol{w}_k$ et $\boldsymbol{w}$ comme à la fin de la section précédente (éq. (4.26)), et l'on fait agir S_k sur F par

$$S_k\cdot y = \sum_{k\geqslant 0} w_{k+1}(z)\,\frac{\log^k z}{k!} = \sum_{n\in\mathbb{Z}}\sum_{k\geqslant 0} y_{n,k+1}\,\frac{\log^k z}{k!}\,z^n.$$

La spécificité de la méthode de Heffter est qu'elle construit toujours les solutions par puissances croissantes de $\log z$. En identifiant $L(z,\theta)$ et S_k à leur action sur F, on pose

$$\mathrm{Sol} = \ker L(z,\theta) = \{y\in F : L(z,\theta)\cdot y = 0\},$$

et l'on introduit pour tout $\sigma\geqslant 0$ l'espace

$$\mathrm{Sol}_\sigma = \mathrm{Sol}\cap(\ker S_k^\sigma)$$

des *solutions de niveau* σ, celles qui ne font intervenir que $\log^0 z, ..., \log^{\sigma-1} z$.

(§56) Heffter observe tout d'abord que si $y(z)$ est solution, alors $S_k \cdot y$ l'est aussi. En effet, on a $L(z,\theta)\cdot y = 0$ si et seulement si $L(z,\theta+S_k)\cdot \boldsymbol{w} = 0$, ce qui implique clairement $L(z,\theta+S_k)\, S_k\cdot \boldsymbol{w} = 0$. **(§57)** Partant de cette observation, il cherche à établir la solution générale de niveau $\sigma+1$ de $L(z,\theta)\cdot y = 0$, connaissant celle de niveau σ. Il s'agit en premier lieu de déterminer les contraintes sur une solution de niveau σ pour qu'elle se « complète » en une solution de niveau $\sigma+1$, autrement dit de caractériser l'espace $\mathrm{Sol}_\sigma \cap S_k\cdot \mathrm{Sol}_{\sigma+1}$. Heffter retient pour cela du système (4.27) la relation

$$L(z,\theta)\cdot w_0 + L'(z,\theta)\cdot w_1 + \cdots + \frac{1}{\sigma!}\, L^{(\sigma)}(z,\theta)\cdot w_\sigma = 0, \tag{4.28}$$

entre les composantes d'une solution y de niveau $\sigma+1$. C'est une équation différentielle inhomogène sur la série w_σ dont le second membre fait intervenir $w_0, \ldots, w_{\sigma-1}$. **(§58)** Passons sur la preuve de convergence des séries w_k ainsi construites.

(§59) Quant aux coefficients $y_{n,k}$ d'une solution de niveau $\sigma+1$, ils vérifient l'équation (4.25) avec $k=\sigma$ — que Heffter obtient en injectant une solution de niveau $\sigma+1$ à coefficients indéterminés dans (4.28). En invoquant le fait que le décalé par S_k d'une solution est lui-même solution, on a donc le système des récurrences (4.25) pour $k = 0, 1, \ldots, \sigma$. **(§60)** Heffter en déduit la condition nécessaire que nous avons déjà vue : il ne peut exister de solution

$$y(z) = (\log^\sigma z + c\log^{\sigma-1} z + \cdots)\, z^n + \tilde{O}(z^{n+1}) \in \mathrm{Sol}_{\sigma+1}\setminus \mathrm{Sol}_\sigma$$

que si n est σ fois racine de Q_0.

(§61) C'est peut-être à se stade que la méthode de Heffter se distingue le plus de ses cousines, en ayant recours à l'isomorphisme

$$(S_k)_*: \quad \mathrm{Sol}_{\sigma+1}/\mathrm{Sol}_1 \xrightarrow{\sim} \mathrm{Sol}_\sigma \cap (S_k\cdot \mathrm{Sol})$$

induit par l'action de S_k restreinte à $\mathrm{Sol}_{\sigma+1}$.

Le procédé pour construire la solution générale de niveau $\sigma+1$ est le suivant. On détermine d'abord l'espace Sol_1 des solutions séries, c'est-à-dire que l'on exprime par récurrence les coefficients d'une série $y\in\mathrm{Sol}_1$ en fonction des y_n pour n racine de Q_0. Supposons Sol_σ construit, et étant donnée une solution $\tilde{y}$ de niveau σ, cherchons les solutions de niveau $\sigma+1$ qui « prolongent » $\tilde{y}$. Il peut se produire deux choses : soit l'équation $S_k\cdot y = \tilde{y}$ n'a pas de solution, soit ses solutions forment un sous-espace affine de $\mathrm{Sol}_{\sigma+1}$ dirigé par Sol_1. La résoudre pour tout $\tilde{y}$ revient à inverser $(S_k)_*$.

Concrètement, on établit à partir des équations (4.25) (pour $k=0$, $t=\sigma$, et n parcourant un intervalle d'entiers qui contient toutes les racines de Q_0) les conditions linéaires sur $\tilde{y}$ pour que $\tilde{y}\in S_k\cdot\mathrm{Sol}$. Lorsque $\tilde{y}$ satisfait ces contraintes, le système des équations (4.25) laisse libres exactement les coefficients de $z^n\log^0 z$ pour $Q(n)=0$, et détermine les autres coefficients de la série w_0 en fonction de ces derniers.

(§62) On itère enfin le procédé pour construire successivement Sol_1, Sol_2, ..., Sol_t. Heffter choisit dans $\mathrm{Sol}_{\sigma+1}\cap(S_k\cdot\mathrm{Sol})$ un supplémentaire de $\mathrm{Sol}_\sigma\cap(S_k\cdot\mathrm{Sol})$, isomorphe à $\mathrm{Sol}_1\cap(S_k^\sigma\cdot\mathrm{Sol})$, et écrit la solution générale de niveau $\sigma+1$ comme la somme de la solution générale de niveau σ et d'une solution de niveau $\sigma+1$ où sont libres certains coefficients de la série w_σ. Les positions des coefficients libres dans w_σ sont un sous-ensemble de celles des coefficients libres dans w_0. Il continue ainsi jusqu'à obtenir r solutions indépendantes. Au final, il ramène donc la détermination d'une solution quelconque à la donnée de r conditions initiales généralisées : les coefficients de $z^n\log^0 z$ pour $Q(n)=0$, suivis de ceux de $z^n\log^k z$ pour des sous-ensembles de racines décroissant avec k. La base ainsi construite est essentiellement notre base canonique, à ceci près que Heffter n'explicite pas que l'on peut choisir les coefficients libres des séries w_k avec $k>1$ de manière à refléter exactement les multiplicités des racines de Q_0.

(§63) Il relie enfin sa méthode à celle de Frobenius, comme suit. Supposons dépassée la plus grande racine de Q_0, et tous les coefficients correspondants déterminés au moyen d'un certain nombre de résolutions de systèmes linéaires. Au vu des

équations (4.25), on peut alors calculer par récurrence les coefficients suivants d'une solution série exprimés en fonction d'une valuation symbolique λ. Avec des conditions initiales (dépendant de λ) convenables, on en déduit les coefficients de la solution générale en dérivant par rapport à λ un nombre de fois égal à la plus grande puissance de $\log z$ attendue, puis en substituant à λ une racine de Q_0.

Chapitre 5

Majorants pour les suites P-récursives

« BORNER, v. act. (*Jardinage.*) du bouis,
par exemple, c'est, lorsqu'il vient d'être planté,
lui donner avec le dos du plantoir ou avec les mains,
la forme & le contour qu'il doit avoir suivant le
dessein, en plombant bien la terre tout au-tour
de peur qu'il ne s'évente. (*K*) »

— Antoine Joseph Dezallier d'Argenville [75, p. 340]

Ce chapitre ainsi que le suivant sont consacrés au calcul de bornes qui contrôlent le comportement de suites P-récursives ou de séries D-finies. Dans ce chapitre, on décrit un algorithme qui prend en entrée la définition d'une suite P-récursive $(u_n)_n$ par une récurrence et des conditions initiales, et renvoie une suite explicite $(v_n)_n$, plus simple que u en un certain sens, telle que $|u_n| \leqslant v_n$ pour tout n. Sur le plan technique, il s'agit de combiner la méthode classique des séries majorantes avec des techniques d'analyse asymptotique comme la méthode du col. Sous une hypothèse de généricité, la majoration obtenue est fine, c'est-à-dire que le comportement asymptotique de v_n quand $n \to \infty$ est proche de celui de u_n, d'une manière contrôlable.

L'essentiel du contenu de ce chapitre est repris d'un article écrit avec Bruno Salvy [175]. Quelques idées (notamment, l'algorithme 5.22 et la proposition 5.34) apparaissent déjà sous forme moins aboutie dans mon rapport de M2 [173].

5.1 Introduction

5.1.1 Problématique

Considérons une suite P-récursive u, définie par une récurrence

$$p^{[s]}(n)\, u_{n+s} + \cdots + p^{[1]}(n)\, u_{n+1} + p^{[0]}(n)\, u_n = 0, \qquad p^{[k]} \in \mathbb{Q}[n], \tag{5.1}$$

accompagnée de conditions initiales appropriées. L'objet de ce chapitre est de décrire un algorithme pour calculer à partir de (5.1) une suite plus simple $(v_n)_n$ telle que $|u_n| \leqslant v_n$ pour tout $n \in \mathbb{N}$. En vertu de la correspondance entre suites P-récursives et séries D-finies, cela revient à trouver une série formelle $v(z) = \sum_n v_n\, z^n$ dont les coefficients majorent terme à terme ceux de la série génératrice $u(z) = \sum_n u_n z^n$ de $(u_n)_n$, qui satisfait quant à elle une équation différentielle linéaire à coefficients polynomiaux. Nous passerons plusieurs fois d'un point de vue à l'autre.

Des bornes de ce genre se révèlent utiles pratiquement dès que l'on veut manipuler les suites P-récursives ou les fonctions D-finies d'un point de vue plus « analytique » que formel et combinatoire. Par exemple, une majoration convenable de u_n peut servir à prouver que la série $\sum_n u_n$ converge, ou à déterminer l'erreur que l'on commet sur sa somme en la tronquant à un certain ordre. Des algorithmes de calcul de bornes qui couvrent le cas des suites P-récursives ou s'y adaptent ont donc été développés dans ce type de contexte, pour certifier la qualité d'approximations numériques [235, 237, 240, 189, 190]. Bien que suffisants pour les applications envisagées, les majorants obtenus sont en général assez grossiers. Ainsi, si $u_n \sim \alpha^n$, la situation typique est que

l'on borne $|u_n|$ par une suite de la forme $A\,\beta^n$ où $\beta > |\alpha|$ peut être pris arbitrairement proche de α — mais quitte à faire croître A de façon incontrôlée.

Or, l'asymptotique des suites P-récursives est par ailleurs un sujet abondamment étudié [194, 94], et le calcul de développement asymptotiques à des ordres arbitraires a été largement automatisé [258, 230, 93, 266, 142]. Nous reviendrons là-dessus en §8.7. Si une estimation asymptotique fournit une indication précise sur le comportement de la suite u_n pour n suffisamment grand, elle ne suffit cependant pas à obtenir une estimation précise pour une valeur de n donnée, ou encore un rang n à partir duquel une inégalité valable asymptotiquement est effectivement réalisée.

L'idée développée ici consiste à marier le calcul d'un majorant avec une analyse (rudimentaire en regard de ce que permet l'asymptotique automatique) du comportement de la suite. Concrètement, il est assez facile dans beaucoup de situations de déterminer, par exemple, que $u_n = O(w_n)$ pour une suite w_n simple et relativement proche de $|u_n|$. La question est de remplacer ce $O(\cdot)$ par une inégalité $|u_n| \leqslant A\,w_n$ avec une constante A explicite. On obtient ainsi des bornes valables pour tout terme u_n, tandis qu'une relation entre le comportement de la borne quand $n \to \infty$ à celui de u garantit que la borne ne s'éloigne pas trop des valeurs qu'elle sert à contrôler. Le résultat principal est le théorème suivant, dont nous préciserons par la suite le vocabulaire.

THÉORÈME 5.1. Soit $(u_n) \in \mathbb{Q}^{\mathbb{N}}$ une suite P-récursive solution de la récurrence homogène (5.1), avec $p^{[s]}(n) \neq 0$ et $p^{[0]}(n) \neq 0$ pour $n \in \mathbb{N}$. Étant données la récurrence (5.1) et des conditions initiales $u_0, ..., u_{s-1}$, l'algorithme 5.30 calcule un réel positif A, un rationnel κ, un nombre algébrique réel positif α et une fonction φ tels que

$$\forall n \in \mathbb{N}, \qquad |u_n| \leqslant A\,n!^{\kappa}\,\alpha^n\,\varphi(n) \tag{5.2}$$

avec $\varphi(n) = e^{o(n)}$. Pour un choix générique des conditions initiales, les paramètres κ et α calculés sont optimaux. ♢

La fonction φ est donnée par une formule explicite, elle-même décrite par un petit nombre de paramètres. Les formes qu'elle peut prendre sont détaillées en §5.4.2.

Dans le cadre d'algorithmes d'évaluation numérique ou d'approximation comme ceux étudiés dans les chapitres suivants de ce mémoire, la finesse des bornes se traduit par un temps de calcul moindre à précision cible égale. Mais comme les majorants sont donnés par des formules explicites dont la forme traduit les propriétés asymptotiques recherchées, ils représentent aussi des résultats « autonomes » dignes d'être renvoyés à l'utilisateur[1]. C'est ainsi que plusieurs bornes obtenues par les algorithmes de ce chapitre et du suivant apparaissent dans le DDMF.

Ce n'est pas un hasard si le critère de finesse du théorème 5.1 correspond à la précision du résultat donné par le théorème de Perron-Kreuser. Ce choix permet de traiter de façon assez uniforme toute la classe des récurrences linéaires à coefficients polynomiaux. À l'inverse, il est possible de raffiner le résultat pour calculer des bornes plus précises en se restreignant à des classes de suites un peu plus contraintes.

Outre le calcul numérique garanti et l'asymptotique automatique, on peut rattacher le contenu de ce chapitre à la thématique des *inégalités* en calcul formel, et particulièrement entre fonctions D-finies [109, 141]. Il faut cependant relever une différence fondamentale : dans un problème d'inégalité, on reçoit en entrée, par exemple, deux fonctions à valeurs réelles f et g, et le but est de décider si $f(x) \leqslant g(x)$ pour tout x. Dans un calcul de majorant, on dispose en revanche d'une certaine liberté dans le choix de g. Ce que l'on peut espérer comme résultats algorithmiques sur le plan inégalités est contraint par des questions de décision profondes, liées notamment au problème de Pisot-Skolem (voir §3.3.4). Ainsi, le meilleur résultat relatif au test de positivité des suites P-récursives — problème naturellement complémentaire de

1. Dans mon expérience jusqu'ici, la motivation des utilisateurs était le plus souvent d'utiliser la borne dans l'analyse d'un algorithme d'évaluation numérique ou de la coder en dur dans l'algorithme.

celui qui nous intéresse ici — est une méthode dont on peut montrer qu'elle termine dans un assez grand nombre de cas [143]. Pour ce qui est du module de suites à valeurs complexes, on pourrait dire que les inégalités se rapprochent plus des bornes *inférieures* que des bornes supérieures, et que les bornes inférieures — non triviales, c'est-à-dire au moins strictement positives ! — sont beaucoup plus difficiles à obtenir[2].

5.1.2 Quelques exemples

Les exemples suivants donnent une idée du genre de bornes que la méthode développée ici permet d'obtenir. Les deux premiers sont empruntés à Wimp et Zeilberger [258, Ex. 2.1 et 2.3], qui illustrent le calcul de développements asymptotiques par la méthode de Birkhoff-Trjitzinsky. Pour des raisons de lisibilité, les constantes qui apparaissent dans les parties polynomiales des résultats ont été remplacées par des approximations par excès à faible précision.

Exemple 5.2. Soit à borner en fonction de $n \in \mathbb{N}$ l'intégrale

$$I_n = \int_0^\infty t^n \, e^{-t^2-1/t} \, \mathrm{d}t.$$

À partir de la relation $2\, I_{n+3} = (n+2)\, I_{n+1} + I_n$ et des inégalités initiales I_0, I_1, $I_2 \leqslant 1/5$, l'algorithme 5.30 détermine que l'on a

$$I_n \leqslant n!^{1/2}\, 2^{-n/2}\, (0{,}26\, n + 0{,}76) \binom{n+19}{19}.$$

On peut montrer qu'en réalité

$$I_n \sim n!^{1/2}\, 2^{-n/2-3/4} \left(\frac{\pi}{n}\right)^{3/4}$$

lorsque $n \to \infty$: les paramètres $\kappa = 1/2$ et $\alpha = 1/\sqrt{2}$ (avec les notations du théorème 5.1) traduisent ce comportement asymptotique. ♢

Exemple 5.3. Supposons que l'on souhaite borner la probabilité qu'une permutation de $[\![1, n]\!]$ tirée uniformément soit une involution. Le nombre t_n d'involutions (oeis A000085 [220]) satisfait la récurrence

$$t_{n+2} = (n+1)\, t_n + t_{n+1}, \qquad t_0 = t_1 = 1,$$

et l'on a

$$t_n \sim (8\,\pi)^{-1/4}\, n!^{1/2}\, e^{\sqrt{n}-1/4}\, n^{-1/4}$$

quand $n \to \infty$ [146, §5.1.4]. Le même algorithme que précédemment conduit à la majoration

$$\frac{t_n}{n!} \leqslant (17575\, n + 52725)\, \frac{1}{\sqrt{n!}} \left(\frac{\sqrt{n+3}}{\sqrt{n+3}-\sqrt{2}}\right)^n e^{\sqrt{2}\sqrt{n+3}} = O\Big(n^{1/4}\, n!^{-1/2}\, e^{2\sqrt{n}}\Big).$$

Observons que sur cet exemple, outre les grandeurs κ et α du théorème 5.1, la croissance sous-exponentielle en $e^{O(\sqrt{n})}$ de la suite $t_n/n!$ est reflétée dans la borne. Cependant, notre algorithme n'est pas conçu pour préserver la constante sous-entendue par ce $O(\cdot)$. En appliquant manuellement une méthode analogue, Flajolet et Sedgewick [94, Ex. VIII.5] arrivent à la borne

$$t_n \leqslant e^{-1/4}\, \sqrt{2\,\pi\, n}\, e^{-n/2+\sqrt{n}}\, n^{n/2}\, (1 + o(1)),$$

un peu plus fine mais, en l'état, valable seulement asymptotiquement. ♢

2. Avec une notion de finesse des majorants plus contraignante que la nôtre, une borne supérieure finirait par induire un encadrement suffisant pour prouver, disons, des résultats de positivité, et on retomberait au final sur les difficultés liées aux inégalités.

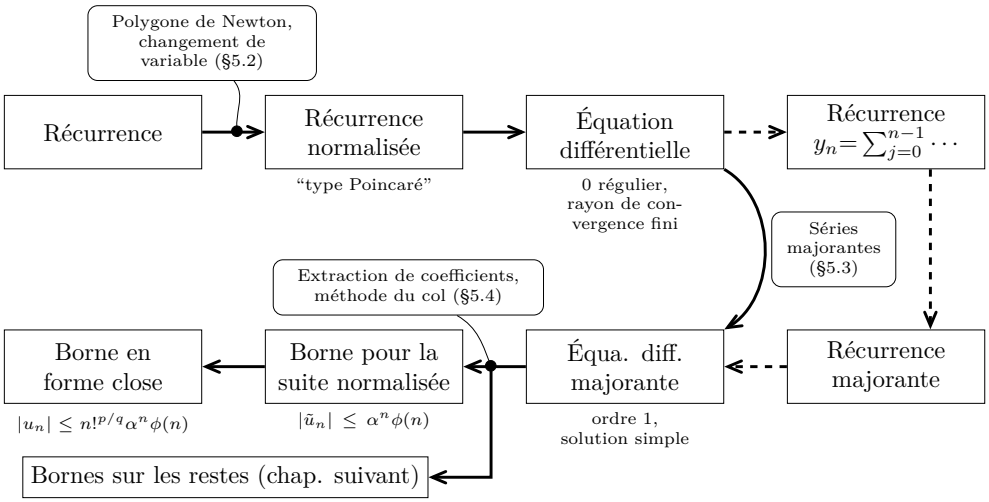


Figure 5.1. Structure des algorithmes de calcul de bornes. Les flèches en trait plein représentent des étapes de calcul, celles en pointillés des étapes de preuve sans contrepartie directe dans l'algorithme.

EXEMPLE 5.4. L'équation différentielle (tirée au hasard)

$$\left(-\frac{7\,z^2}{10}-\frac{7\,z}{10}-\frac{2}{3}\right)y^{(3)}(z)+\left(\frac{3\,z^2}{2}-\frac{z}{6}-\frac{43}{60}\right)y''(z)$$
$$+\left(-\frac{7\,z^2}{20}+\frac{8\,z}{15}+\frac{47}{60}\right)y'(z)+\left(\frac{4\,z^2}{15}-\frac{z}{3}-\frac{1}{60}\right)y(z)=0$$

admet une base de solutions analytiques sur le disque centré à l'origine de rayon $\rho = 2\sqrt{105}/21$. À partir de cette équation, l'algorithme 5.26 établit que le développement en série sur ce disque de la solution correspondant aux conditions initiales

$$y(0)=-\frac{3}{20},\quad y'(0)=\frac{4}{5},\quad y''(0)=-\frac{13}{30}$$

est borné coefficient par coefficient par celui de la fraction rationnelle

$$\frac{13250086}{\left(1-\dfrac{z}{\rho}\right)^2},$$

elle-même analytique sur le disque de rayon ρ. Nous en déduirons au chapitre suivant des bornes sur les restes $y_{n;}(z)$ du développement en série de y, utilisables entre autres pour évaluer y à précision garantie. ♢

5.1.3 Esquisse de l'algorithme et plan du chapitre

La figure 5.1 donne les grandes lignes de l'algorithme de calcul de majorants et de la suite du chapitre. Considérons une solution (u_n) de la récurrence (5.1).

§5.2. On commence par étudier le polygone de Newton de la récurrence et les équations caractéristiques associées pour délimiter, à l'aide du théorème de Perron-Kreuser, les comportements asymptotiques possibles pour u_n. Par une transformation simple de l'opérateur, on « met en facteur » la partie dominante du comportement asymptotique, et l'on est ramené à majorer le facteur restant $\tilde{u}_n$, solution d'une « récurrence normalisée ». Grâce à la correspondance entre suites P-récursives et fonctions D-finies, on code $\tilde{u}$ par sa série génératrice, que l'on sait analytique à l'origine et solution d'une équation différentielle dont l'origine est un point ordinaire ou singulier régulier. Le comportement asymptotique obtenu à cette étape est représenté par deux réels notés κ et α.

§5.3. On adapte ensuite la méthode des séries majorantes pour borner cette série génératrice à partir de séries majorantes sur les coefficients de l'équation différentielle. Le point clé à cette étape est de trouver une série majorante dont le disque de convergence s'étende jusqu'à la plus proche singularité non nulle de l'équation différentielle. Côté borne sur les coefficients, on échappe ainsi à la perte d'un facteur exponentiel habituellement associée à la méthode des séries majorantes. La série majorante calculée est complètement déterminée par trois paramètres T, K et A.

§5.4. Enfin, on combine les résultats en un vecteur $(\kappa, \alpha, T, K, A)$ qui décrit la borne cherchée. On étudie la classe des séries majorantes possibles pour donner en fonction des paramètres les formules explicites annoncées dans le théorème, ce qui en termine la preuve.

La section 5.5 décrit brièvement l'implémentation. La section 5.6 conclut le chapitre avec quelques remarques à propos des variantes possibles et des limitations de l'algorithme.

Notation 5.5. Dans ce chapitre, la notation y_n où $y \in \mathbb{C}[[z]]$ désigne exclusivement le coefficient de z^n dans y. On utilise des exposants entre crochets comme dans $p^{[0]}$ pour noter les autres indices quand il y a risque d'ambiguïté. ♢

5.2 Paramètres de croissance factorielle et exponentielle

Les parties « factorielle » en $n!^\kappa$ et « exponentielle » en α^n du comportement asymptotique d'une solution sont celles préservées par notre algorithme. La première étape de l'algorithme consiste à extraire, à ce niveau de précision, le comportement d'une solution générique.

5.2.1 Singularités dominantes

Définition 5.6. Si $P \in \mathbb{Q}[z]$ est un polynôme non réduit à un monôme, appelons provisoirement δ-racines de P celles de multiplicité maximale parmi ses racines non nulles de module minimal. On note respectivement

$$\delta(P) = \min\{|\zeta| \neq 0 : P(\zeta) = 0\} \quad \text{et} \quad \nu_\delta(P) = \max\{\nu(\zeta, P) : |\zeta| = \delta(P)\}$$

le module et la multiplicité des δ-racines de P. On convient en outre qu'un monôme a une δ-racine infinie (de multiplicité indéfinie). On appelle *pôles dominants* d'une fraction rationnelle les δ-racines de son dénominateur, et *singularités dominantes* d'un opérateur différentiel à coefficients polynomiaux celles de son coefficient de tête. ♢

Les $\delta(P)$ pour $P \in \mathbb{Q}[z]$ forment un sous-ensemble stable par multiplication des nombres algébriques réels strictement positifs. En complément d'outils standard de manipulation symbolique, nous supposons disposer des primitives suivantes, qui opèrent sur cette représentation.

Approximation par défaut. Nous supposons l'existence d'une fonction qui calcule des approximations numériques par défaut à précision arbitraire de $\delta(P)$. En pratique, les solveurs polynomiaux modernes (par exemple MPSolve [27] ou ceux intégrés aux systèmes de calcul formel majeurs) fournissent la fonctionnalité d'évaluation numérique demandée, et bien plus encore. Dans la mesure où nous ne nous intéressons qu'à $\delta(P)$ et non à l'ensemble des racines de P, il est aussi possible d'écrire une procédure *ad hoc* simple fondée sur la méthode de Gräffe [217, §14].

Comparaison. Il s'agit de décider, étant donnés P et Q, si l'on a $\delta(P) < \delta(Q)$, $\delta(P) = \delta(Q)$ ou $\delta(P) > \delta(Q)$. On peut par exemple recourir à une approche symbolique-numérique simple dans l'esprit de celle de Gourdon et Salvy [114].

De façon plus générale, les étapes des algorithmes 5.22 et 5.26 qui ne mentionnent aucune précision cible peuvent être exécutées en arithmétique d'intervalles, voire simplement en arithmétique flottante (avec un peu de soin sur les modes d'arrondi) plutôt que symboliquement. Nous reviendrons sur ce point au chapitre suivant, en §6.3.2.

REMARQUE 5.7. Les résultats de ce chapitre sont énoncés pour des équations à coefficients rationnels, mais s'adaptent sans difficulté à n'importe quel sous-corps $\mathbb{K}$ « suffisamment effectif » de $\mathbb{C}$. En particulier, nous utiliserons la variante $\mathbb{K} = \mathbb{Q}[i]$ dans le contexte du prolongement analytique numérique. La façon de réaliser les opérations sur $\delta(P)$ ainsi que quelques détails des algorithmes (essentiellement de l'algorithme 5.22) dépendent de $\mathbb{K}$. ♢

5.2.2 Croissance générique des solutions

ALGORITHME 5.8. Asympt(R)

Entrée. $R = \sum_{k=0}^{s} b^{[k]}(n)\, S^k \in \mathbb{Q}[n]\langle S\rangle$.

Sortie. $\kappa \in \mathbb{Q}$, $P_\alpha \in \mathbb{Q}[z]$.

1 $\kappa := \max_{k=0}^{s-1} \dfrac{\deg b^{[k]} - \deg b^{[s]}}{s-k}$

2 $P_\alpha := \sum_{\ell=0}^{s} b_{d+\ell\kappa}^{[s-\ell]}\, z^\ell$ où $d = \deg b^{[s]}$

3 renvoyer (κ, P_α) ♢

Soit $R \in \mathbb{Q}[n]\langle S\rangle$ un opérateur réversible, non singulier, d'ordre s. Une solution $u = (u_n)_{n\in\mathbb{N}}$ de la récurrence $R \cdot u = 0$ est alors déterminée de façon unique par ses premières valeurs $u_0, ..., u_{s-1} \in \mathbb{C}$. Nous dirons qu'une assertion est vraie pour une solution générique, ou pour des conditions initiales génériques, si elle est satisfaite pour $(u_0, ..., u_{s-1}) \in \mathbb{C}^s \setminus V$ où V est un sous-espace vectoriel strict de $\mathbb{C}^s$.

D'après le théorème de Perron-Kreuser (théorème 3.25), les comportements « factoriel » et « exponentiel » des solutions « dont la croissance est la plus rapide » sont déterminés respectivement par la pente de l'arête la plus à droite du polygone de Newton de R et les racines de module maximal de son équation caractéristique (elles-mêmes liées aux singularités dominantes de l'équation différentielle associée). L'algorithme 5.8 extrait cette information, qui correspond en fait au comportement asymptotique d'une solution générique de $R \cdot u = 0$, d'après la proposition 5.9.

PROPOSITION 5.9. Posons $R = \sum_{k=0}^{s} b^{[k]}(n)\, S^k \in \mathbb{Q}[n]\langle S\rangle$ et supposons que R n'est pas réduit à un terme $b^{[s]}\, S^s$. L'algorithme 5.8 calcule un couple $(\kappa, P_\alpha) = \mathrm{Asympt}(R)$ tel que

$$\limsup_{n\to\infty} \left|\frac{u_n}{n!^\kappa}\right|^{1/n} \leqslant \alpha \qquad \text{où } \alpha = \frac{1}{\delta(P_\alpha)}$$

pour toute solution (u_n) de $R \cdot u = 0$, avec égalité pour une solution générique. ♢

DÉMONSTRATION. L'arête la plus à droite du polygone de Newton a pour pente $-\kappa$ et l'équation caractéristique associée est le polynôme réciproque de P_α, d'où l'inégalité d'après le théorème 3.25. Il reste à démontrer que l'on a égalité pour des conditions initiales génériques. Soit $V = \ker R \subset \mathbb{C}^{\mathbb{N}}$. À nouveau d'après le théorème 3.25, il existe $u^{[1]} \in V$ telle que

$$\limsup_{n\to\infty} \left|\frac{u_n^{[1]}}{n!^\kappa}\right|^{1/n} = \alpha.$$

Étendons une telle solution en une base $(u^{[1]}, ..., u^{[s]})$ de V. Soit $u = \sum_k \lambda^{[k]}\, u^{[k]} \in V$. Par construction de κ et α, on a $\limsup_{n\to\infty} |u_n/n!^\kappa|^{1/n} \leqslant \alpha$. Quitte à extraire des

sous-suites, on peut supposer que $u_n^{[1]}$ n'est jamais nul, que $u_n^{[1]}/n!^\kappa \to \alpha$ quand $n\to\infty$, et qu'il existe $\beta \leqslant \alpha$ (éventuellement nul) tel que $u_n/n!^\kappa \to \beta$ quand $n\to\infty$. On a alors

$$\left|\lambda^{[1]} + \lambda^{[2]}\frac{u_n^{[2]}}{u_n^{[1]}} + \cdots + \lambda^{[s]}\frac{u_n^{[s]}}{u_n^{[1]}}\right|^{1/n} \underset{n\to\infty}{\longrightarrow} \frac{\beta}{\alpha},$$

et $\beta = \alpha$ à moins que

$$\frac{\lambda^{[2]}\,u_n^{[2]} + \cdots + \lambda^{[s]}\,u_n^{[s]}}{u_n^{[1]}} \underset{n\to\infty}{\longrightarrow} -\lambda^{[1]},$$

condition fausse pour des $\lambda^{[k]}$ génériques. □

Définition 5.10. Nous appelons *arête dominante* l'arête la plus à droite du polygone de Newton (au sens de la définition 3.24), *équation caractéristique dominante* l'équation caractéristique associée à l'arête dominante, et *récurrence normalisée* une relation (ou un opérateur) de récurrence pour lequel cette arête est horizontale. ♢

Une récurrence est donc normalisée lorsque ses solutions « de croissance la plus rapide » ont un comportement asymptotique purement exponentiel, par opposition à factoriel.

5.2.3 Fonction génératrice normalisée

Algorithme 5.11. RecToDiffeq(R)

Entrée. Un opérateur $R = \sum_{k=0}^s b^{[k]}\,S^k \in \mathbb{Q}[n]\langle S\rangle$.

Sortie. Un opérateur $D \in \mathbb{Q}[z]\langle\theta\rangle$ tel que $(u_n)_{n\in\mathbb{N}} \in \ker R \Rightarrow D\cdot\sum_{n\in\mathbb{N}} u_n\,z^n = 0$.

1 $g := \Pi/\mathrm{pgcd}\big(b^{[s]}, \Pi\big)$ où $\Pi = \prod_{k=1}^s (n+k)$

2 calculer les coefficients $c_{k,j}$ de $g\,R = \sum_{k=0}^s c_{k,j}\,n^j\,S^k$
[on a ainsi $R = \sum_{k=0}^s c_{k,j}\,S^k\,(n-k)^j$*]*

3 développer $\sum_{k=0}^s \sum_j c_{k,j}\,z^{s-k}\,(\theta-k)^j$ sous la forme $D = \sum_{k=0}^r a^{[k]}\,\theta^k$

4 renvoyer D ♢

Considérons à nouveau un opérateur non singulier $R = \sum_{k=0}^s b^{[k]}\,S^k \in \mathbb{Q}[n]\langle S\rangle$, avec $b^{[0]}$, $b^{[s]} \neq 0$. Nous avons vu (§3.3.2) que la série génératrice $u(z)$ d'une suite $u \in \ker R$ est annulée par l'opérateur $D = \sum_{k=0}^r a^{[k]}\,\theta^k \in \mathbb{Q}[z]\langle\theta\rangle$ calculé par la fonction RecToDiffeq de l'algorithme 5.11. La multiplication par g ligne 1 provient de ce qu'on fait agir l'opérateur R sur des suites indexées par les entiers *positifs*, alors que l'opérateur $\sum_k b^{[k]}(\theta)\,z^{-k}$ annule les séries génératrices des suites satisfaisant $R\cdot u = 0$ indexées par $\mathbb{Z}$ (cf. §3.2.1). En divisant par $a^{[r]}$, on obtient encore

$$\left(\theta^r + \frac{a^{[r-1]}}{a^{[r]}}\,\theta^{r-1} + \cdots + \frac{a^{[1]}}{a^{[r]}}\,\theta + \frac{a^{[0]}}{a^{[r]}}\right)\cdot u = 0. \tag{5.3}$$

Lemme 5.12. Si l'opérateur R est normalisé au sens de la définition 5.10, alors l'origine est un point régulier de $D = \mathrm{RecToDiffeq}(R)$, et l'équation caractéristique dominante de R est le polynôme réciproque de $a^{[r]}$. ♢

Démonstration. On reprend les notations de la fonction RecToDiffeq, et l'on pose $m = \deg g$ et $d^{[k]} = \deg b^{[k]}$ pour tout k. On a donc $r = m + \max_{k=0}^s d^{[k]}$. Le terme dominant de $\theta^j\,z^{-k}$ vu comme un opérateur en θ à coefficients polynômes de Laurent en z est $z^{-k}\,\theta^j$, d'où $a^{[r]}(z) = \sum_{k=0}^s c_{k,r}\,z^{s-k}$. La condition de normalisation de R se traduit par $d^{[s]} = \max_{k=0}^{s-1} d^{[k]}$, c'est-à-dire $d^{[k]} = d^{[s]} = r - m$ pour un certain $k < s$. Ainsi, on a $a^{[r]}(0) = c_{s,r} \neq 0$, et d'après le critère de Fuchs (théorème 4.8), l'origine

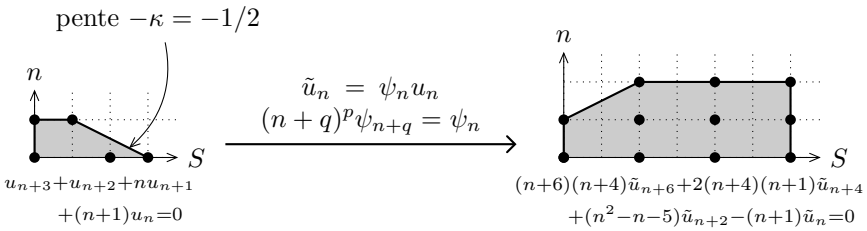


Figure 5.2. Polygone de Newton d'un opérateur de récurrence, avant et après normalisation.

est un point régulier de D. Enfin, l'expression générale de l'équation caractéristique dominante,

$$\chi_E(\lambda) = \lambda^{-t} \sum_{\substack{d^{[k]}+k\kappa(E) \\ = d^{[s]}+s\kappa(E)}} a_{k,d^{[k]}}\,\lambda^k$$

où t est choisi de sorte que $\chi_E(0) \neq 0$, se simplifie ici en

$$\chi_E(\lambda) = \lambda^{-t} \sum_{d^{[k]}=r} a_{k,r}\,\lambda^k. \qquad \square$$

Nous verrons dans la section suivante comment borner finement les solutions d'une récurrence normalisée *via* celles de l'équation différentielle associée. Dans le cas général, on commence par normaliser R par un changement de variable qui préserve la nature polynomiale des coefficients. La figure 5.2 en donne un exemple et illustre l'action de la normalisation sur les polygones de Newton.

L'algorithme 5.13 réalise la normalisation. Le produit symétrique à l'étape 2 n'est pas quelconque. Je renvoie à Barkatou *et al.* [12] pour différentes façons de le calculer.

ALGORITHME 5.13. Normalize(R, κ)

Entrée. Un opérateur de récurrence $R \in \mathbb{Q}[n]\langle S\rangle$. Un rationnel κ.

Sortie. Un opérateur différentiel $D \in \mathbb{Q}[z]\langle\theta\rangle$.

1 $p/q := \kappa$, avec $p \in \mathbb{Z}$, $q \in \mathbb{N}^*$ et $\operatorname{pgcd}(p, q) = 1$

2 calculer les coefficients $\hat{b}^{[k]}(n)$ du produit symétrique $\hat{R} = \sum_{k=0}^{qs} \hat{b}^{[k]}(n)\,S^k$ de R par $(n+q)^p\,S^q - 1$

3 renvoyer RecToDiffeq$\big(\hat{R}\big)$ (algorithme 5.11) $\diamondsuit$

PROPOSITION 5.14. Soit $R \in \mathbb{Q}[n]\langle S\rangle$ un opérateur non singulier, réversible, de coefficient constant par rapport à S non nul. Soit $(p/q, P_\alpha)$ le comportement asymptotique générique des éléments de $\ker R$ tel que renvoyé par l'algorithme 5.8 ; on suppose que $\delta(P_\alpha) < \infty$. Alors l'algorithme 5.13 calcule un opérateur différentiel $D =$ Normalize$(R, p/q)$ qui annule la série

$$\tilde{u}(z) = \sum_{n=0}^{\infty} \psi_n\,u_n\,z^n$$

pour toutes suites ψ et u solutions respectivement de

$$(n+q)^p\,\psi_{n+q} = \psi_n \tag{5.4}$$

et $R \cdot u = 0$. L'opérateur D est régulier à l'origine et le module de sa singularité dominante est égal $\delta(P_\alpha)$. $\diamondsuit$

DÉMONSTRATION. Soit $(u^{[1]}, \ldots, u^{[s]})$ une base du noyau de R avec les comportements asymptotiques donnés par le théorème de Perron-Kreuser, c'est-à-dire

$$\operatorname*{limsup}_{n\to\infty} \left|\frac{u_n^{[k]}}{n^{\kappa_k}}\right|^{1/n} = \alpha_k$$

où les (κ_k, α_k) se lisent sur le polygone de Newton. En particulier, on a

$$\limsup_{n\to\infty} \left| \frac{u_n^{[k]}}{n^{p/q}} \right|^{1/n} \leqslant \alpha$$

pour tout k. Notons $(\psi^{[0]}, \ldots, \psi^{[q-1]})$ la base de solutions de la récurrence (5.4) déterminée par les conditions initiales $\psi_j^{[i]} = \mathbb{1}[i=j]$ pour $0 \leqslant i, j < q$. L'algorithme 5.13 construit un opérateur $\hat{R}$ tel que, pour N grand, les $s\,q$ suites $(\psi_n^{[j]}\, u_n^{[k]})_{n\geqslant N}$ engendrent $\{\hat{u} \mid \forall n \geqslant N, (\hat{R}\cdot\hat{u})_n = 0\}$. On a

$$\forall j, \forall k, \qquad \limsup_{n\to\infty} \left| \psi_n^{[j]}\, u_n^{[k]} \right|^{1/n} \leqslant \alpha.$$

Supposons que

$$\hat{u} = \sum_{j,k} \lambda^{[j,k]}\, \psi^{[j]}\, u^{[k]}$$

satisfait $(\hat{R}\cdot\hat{u})_n = 0$ au voisinage de l'infini. Il s'ensuit que $\limsup |u_n|^{1/n} \leqslant \alpha$ (en effet, pour tout $\varepsilon > 0$, on a $|u_n| \leqslant (\alpha+\varepsilon)^n$ pour n grand). Par ailleurs, il existe (j,k) tel que

$$\limsup_{n\to\infty} |\psi_n^{[j]}\, u_n^{[k]}|^{1/n} = \alpha.$$

D'après le théorème de Perron-Kreuser, l'opérateur $\hat{R}$ est normalisé et le module maximal d'une racine de son équation caractéristique dominante est égal à α. Le lemme 5.12 traduit cette propriété sur l'opérateur D, ce qui conclut la preuve. □

Il sera commode pour la suite de choisir une solution explicite de (5.4) à utiliser dans les normalisations. Nous adoptons

$$\psi_n = q^{-\frac{p}{q}n}\, \Gamma\!\left(\frac{n}{q}+1\right)^{-p}. \tag{5.5}$$

Lemme 5.15. La suite $(\psi_n)_{n\in\mathbb{N}}$ est monotone pour tout $p/q \in \mathbb{Q}$. ♢

Démonstration. Pour $q=1$, la définition (5.5) se simplifie en $\psi_n = n!^{-p}$. Supposons $q \geqslant 2$, et posons $f(x) = q^x\, \Gamma(x+1)$. On a

$$\frac{f'(x)}{f(x)} = \ln q + \digamma(x+1) \geqslant 0, \qquad x \geqslant 0,$$

puisque la fonction digamma $\digamma = \Gamma'/\Gamma$ est croissante sur $]0, +\infty[$ et satisfait $\digamma(1) = -\gamma \geqslant -\ln 2$, où γ désigne la constante d'Euler-Mascheroni. La suite (ψ_n) est donc croissante si $p < 0$ et décroissante si $p > 0$. □

Remarque 5.16. La transformation $u \mapsto \tilde{u}$ peut être vue comme une transformée de Borel généralisée [255]. Pour $p = -1$, on a

$$q^{\frac{n}{q}}\, \Gamma\!\left(\frac{n}{q}+1\right) = \int_0^\infty \exp\left(-\frac{t^q}{q}\right) t^{n+q-1}\, \mathrm{d}t,$$

d'où, si u est une fonction dont la croissance des coefficients est caractérisée par les paramètres $\kappa = -1/q$ et α,

$$\begin{aligned}
\tilde{u}(z) &= \sum_{n=0}^{\infty} \psi_n\, u_n\, z^n \\
&= \sum_{n=0}^{\infty} \left(\int_0^\infty \exp\left(-\frac{t^q}{q}\right) t^{n+q-1}\, \mathrm{d}t \right) u_n\, z^n \\
&= \int_0^\infty t^{q-1} \exp\left(-\frac{t^q}{q}\right) u(t\,z)\, \mathrm{d}t.
\end{aligned}$$

(La dernière intégrale converge pour $|z| < \alpha^{-1}$ car $|f(z)| \leqslant \exp\left((q^{-1}\,\alpha + o(1))\,|z|^q\right)$, voir §6.2.2.) La transformée inverse est donnée par

$$u(z) = \oint_\gamma \bar{\psi}\Big(\frac{z}{w}\Big)\,\frac{\tilde{u}(w)}{w}\,\mathrm{d}\,w, \qquad \bar{\psi}\,(z) = \sum_{n=0}^{\infty} \psi_n^{-1}\,z^n,$$

où γ est un contour entourant l'origine et situé dans le disque de convergence de $\tilde{u}$. La fonction $\bar{\psi}$ s'écrit explicitement en termes de fonctions hypergéométriques généralisées ; hélas, cela ne suffit pas à exprimer simplement $u(z)$ dans les cas intéressants. ♢

5.3 Contrôle du comportement sous-exponentiel

Les résultats de la section précédente fournissent une manière de calculer les comportements factoriel et exponentiel génériques des solutions d'une récurrence linéaire à coefficients polynomiaux, et d'extraire les facteurs correspondants d'une description de ces solutions. L'étape suivante consiste à *borner* le facteur sous-exponentiel restant, cette fois en tenant compte des conditions initiales.

5.3.1 La méthode des séries majorantes

Notre outil principal pour ce faire est une variante de la *méthode des séries majorantes*, ou méthode de Cauchy-Kovalevskaya. Technique classique pour établir la convergence de solutions d'équations différentielles obtenues comme séries formelles [128, Chap. 2], elle a aussi occasionnellement été utilisée pour obtenir des bornes explicites sur les restes de ces séries ou d'autres quantités liées (voir §6.1.2). L'originalité de notre approche réside dans l'objectif de finesse des bornes.

DÉFINITION 5.17. Une série formelle $v \in \mathbb{R}_+[[z]]$ est appelée *série majorante* de $u \in \mathbb{C}[[z]]$ si v domine u coefficient par coefficient, c'est-à-dire si $\forall n, |u_n| \leqslant v_n$. On note alors $u \preccurlyeq v$. On étend la définition aux matrices, coefficient par coefficient. ♢

Le lemme suivant collecte quelques propriétés immédiates de compatibilité entre la relation de majoration et les opérations usuelles sur les séries.

LEMME 5.18. Soient $u, u^{[1]}, u^{[2]} \in \mathbb{C}[[z]]$ et $v, v^{[1]}, v^{[2]} \in \mathbb{R}_+[[z]]$ des séries telles que $u \preccurlyeq v, u^{[1]} \preccurlyeq v^{[1]}, u^{[2]} \preccurlyeq v^{[2]}$. Alors

(a) le disque de convergence de v est inclus dans celui de u ;

(b) si ζ appartient au disque de convergence de v, alors $|u(\zeta)| \leqslant v(|\zeta|)$;

(c) pour tous $m, n \in [\![0, \infty]\!]$, on a $u_{m;n} \preccurlyeq v_{m;n}$;

(d) on a les majorations

$$u' \preccurlyeq v' \quad ; \quad u^{[1]} + u^{[2]} \preccurlyeq v^{[1]} + v^{[2]} \quad ; \quad u^{[1]}\,u^{[2]} \preccurlyeq v^{[1]}\,v^{[2]}\,;$$

(e) si en outre $v^{[1]}(0) = 0$, on a $u^{[2]} \circ u^{[1]} \preccurlyeq v^{[2]} \circ v^{[1]}$. ♢

L'idée est de contrôler l'équation étudiée donnée par une *équation majorante* dont les solutions sont automatiquement des séries majorantes de celles de l'équation majorée, dès lors que certaines inégalités sur les conditions initiales sont satisfaites. Il s'agit en quelque sorte d'une version rigoureuse de la règle heuristique des « fonctions modèles » [36, Chap. 2], qui préconise, pour le calcul d'une fonction inconnue, de régler les paramètres de précision d'une façon qui conviendrait dans le cas d'une fonction connue dont on s'attend à ce qu'elle présente un comportement analytique comparable.

Commençons par illustrer le fonctionnement de la méthode des séries majorantes sur un exemple simple.

Exemple 5.19. Soit a une fonction analytique sur $D=\{z,|z|<\rho\}$. Alors toute série formelle u solution de $u'(z)=a(z)\,u(z)$ converge sur D. ♢

Démonstration. Posons $a(z)=\sum_{n=0}^{\infty} a_n\,z^n$ et cherchons u sous la forme $u(z)=\sum_{n=0}^{\infty} u_n\,z^n$. En injectant ces formes à coefficients indéterminées dans l'équation et en extrayant le coefficient de z^n, on a $(n+1)\,u_{n+1}=\sum_{j=0}^{n} a_j\,u_{n-j}$. Soient $\gamma<\rho$ et M tel que $\forall j, |a_j|\leqslant M\gamma^{-j}$. Considérons la récurrence $(n+1)\,v_n=\sum_{j=0}^{n} M\gamma^{-j}\,v_{n-j}$. La série génératrice $v(z)=\sum_{n=0}^{\infty} v_n\,z^n$ d'une solution satisfait

$$v'(z)=\frac{M}{1-\gamma^{-1}\,z}\,v(z)$$

d'où $v(z)\propto(1-\gamma^{-1}\,z)^{-M\gamma}$. Clairement, $|u_0|\leqslant v_0$ implique $u(z)\preccurlyeq v(z)$, or v est analytique sur le disque $|z|<\gamma$, donc u aussi. On obtient le résultat en faisant tendre γ vers ρ. □

L'équation majorante est souvent, mais pas nécessairement, de même « forme » que celle de départ. Notamment, si l'on a

$$\begin{cases} u^{(r)}=a^{[r-1]}\,u^{(r-1)}+\cdots+a^{[0]}\,u \\ v^{(r)}=b^{[r-1]}\,v^{(r-1)}+\cdots+b^{[0]}\,v \end{cases} \qquad |u(0)|\leqslant v(0),\ldots,|u^{(r-1)}(0)|\leqslant v^{(r-1)}(0) \tag{5.6}$$

où les $a^{[k]}$, $b^{[k]}$ sont analytiques en zéro et $a^{[k]}\preccurlyeq b^{[k]}$ pour tout k, alors $u\preccurlyeq v$ par le même raisonnement que dans l'exemple 5.19. L'argument ne tient plus si l'une des fonctions $a^{[k]}$ présente un pôle en zéro ; cependant la méthode des majorants se généralise assez directement au cas d'un point singulier régulier (voir Mezzino et Pinsky [176] pour un exemple détaillé, et van der Hoeven [237, Prop. 3.7] pour un énoncé général).

Nous étudions maintenant une généralisation de ce type, conçue pour satisfaire les contraintes suivantes. Premièrement, la méthode doit être complètement algorithmique. Deuxièmement, elle doit s'appliquer au cas d'un point singulier régulier quelconque[3]. Troisièmement, et surtout, elle doit produire des bornes *fines*. Le critère de finesse du théorème 5.1 se traduit comme suit.

Définition 5.20. Soit u une série entière de rayon de convergence fini non nul. Nous dirons que v est une série majorante *fine* de u, ou majore finement u, lorsque $u\preccurlyeq v$ et v a le même rayon de convergence que u. ♢

Comme dans l'exemple 5.19 ou l'équation (5), la première étape consiste à calculer des séries majorantes des *coefficients* de l'équation, qui dans le cas qui nous intéresse sont des fractions rationnelles.

5.3.2 Séries majorantes pour les fractions rationnelles

Soit $r(z)=N(z)/D(z)=\sum_{n=0}^{\infty} r_n\,z^n$ une fraction rationnelle sans pôle en zéro. En décomposant r en éléments simples sur les complexes, on écrit le coefficient général du développement en série de r sous la forme

$$r_n=\sum_{D(\zeta)=0}\ \sum_{d=1}^{\nu(\zeta,D)} h^{[\zeta,d]}\,(n+1)^{\uparrow(d-1)}\,\zeta^{-n}, \quad n\geqslant\max(0,\deg N-\deg D+1) \tag{5.7}$$

où les $h^{[\zeta,d]}\in\mathbb{Q}(\zeta)$.

Notre but est de calculer une série majorante de r dont le pôle dominant ait même module et même ordre que celui de r. En termes de coefficients, il s'agit donc d'obtenir une borne de la forme $|r_n|\leqslant M\,\delta(D)^{-n}\,n^{\nu_\delta(D)}$. Dans l'optique de la suite,

3. Pour l'application aux suites P-récursives, seules les solutions séries nous intéressent. Plus loin (§6.4), nous généraliserons cette méthode pour borner les solutions séries généralisées en un point singulier régulier présentées en §3.3.1.

l'algorithme prend en entrée un polynôme P_α et un entier $m \geqslant 1$ tels que $\delta(P_\alpha) \geqslant \delta(D)$ et $\delta(P_\alpha) = \delta(D) \Rightarrow m \geqslant \nu_\delta(D)$. Il renvoie une constante M telle que

$$r(z) \trianglelefteq \frac{M}{(1-\alpha\, z)^m}, \qquad \alpha = \frac{1}{\delta(P_\alpha)}. \tag{5.8}$$

Pour calculer un M convenable, on part du membre droit de (5.7) divisé par

$$b_n = [z^n]\, \frac{1}{(1-\alpha z)^m} = (n+1)^{\uparrow(m-1)}\, \alpha^n.$$

Par inégalité triangulaire, le quotient r_n/b_n est borné par une somme $t(n)$ de termes de la forme

$$c\, \frac{(n+1)^{\uparrow(d-1)}}{(n+1)^{\uparrow(m-1)}}\, \lambda^n, \qquad c \geqslant 0, \quad 0 < \lambda \leqslant 1,$$

avec en outre $d \leqslant m$ lorsque $\lambda = 1$. Chacun de ces termes est décroissant pour $n \geqslant 1$ si $d \leqslant m$, et pour $n \geqslant (d-m)/\log(1/\lambda)$ sinon. On calcule un indice N_0 à partir duquel la suite $t(n)$ décroît et satisfait l'inégalité $|r_n/b_n| \leqslant t(n)$, puis on ajuste M à partir de $t(N_0)$ et des valeurs explicites des premiers coefficients de r.

Détaillons le calcul de $t(n)$. On considère la décomposition sans carré

$$D = D_1\, D_2^2 \cdots D_k^k. \tag{5.9}$$

Via la décomposition en éléments simples de r relative à la factorisation partielle (5.9), on observe que le coefficient $h^{[\zeta,d]}$ apparaissant dans (5.7) s'écrit $h^{[\zeta,d]} = \zeta^{-d}\, h_{i,d}(\zeta)$, où $h_{i,d} \in \mathbb{Q}[X]$ dépend de d et du polynôme D_i tel que $D_i(\zeta) = 0$, mais pas du choix de ζ parmi les racines de D_i. Cette écriture est à la base de l'algorithme de Bronstein-Salvy pour le calcul de la décomposition en éléments simples sur $\mathbb{C}$ sans factorisation de polynômes [43] (voir aussi Bronstein [42, §2.7]). En majorant $|\zeta|^{-1}$ par $\delta(D_i)^{-1}$, il vient

$$\begin{aligned}\left|\frac{r_n}{b_n}\right| &= \left|\alpha^{-n} \sum_{i=1}^{k} \sum_{D_i(\zeta)=0} \sum_{d=0}^{i-1} \zeta^{-d}\, h_{i,d}(\zeta)\, \frac{(n+1)^{\uparrow(d-1)}}{(n+1)^{\uparrow(m-1)}}\, \zeta^{-n}\right| \\ &\leqslant \sum_{i=1}^{k} \sum_{d=0}^{i-1} \left(\sum_{D_i(\zeta)=0} |\zeta^{-d}\, h_{i,d}(\zeta)| \right) \frac{(n+1)^{\uparrow(d-1)}}{(n+1)^{\uparrow(m-1)}}\, (\alpha\, \delta(D_i))^{-n}. \end{aligned} \tag{5.10}$$

On adopte comme borne $t(n)$ le membre droit de (5.10) ou une approximation numérique par excès de celui-ci. Pour traiter la somme entre parenthèses, on peut borner les $\zeta^{-d}\, h_{i,d}(\zeta)$ en remplaçant à nouveau chaque ζ^ℓ par $\delta(D_i)^\ell$ ou $\delta(\mathrm{Rcpq}\, D_i)^\ell$ suivant le signe de ℓ. Bien souvent cependant, il est plus simple d'encadrer toutes les racines de D_i à l'aide d'un solveur polynomial garanti et de calculer la somme en arithmétique d'intervalles.

REMARQUE 5.21. Cette façon de procéder n'est pas optimale. Déjà en adoptant des majorants de la forme (5.8), on perd une partie de l'information nécessaire pour rendre l'erreur de la borne finale polynomiale en n. Une approche plus fine prendrait soin de préserver les coefficients de tous les éléments simples de r correspondant aux pôles dominants. Il y a cependant un compromis à faire entre finesse et simplicité des majorants. Ceux-ci suffisent pour nos besoins, et dans l'application à venir aux fonctions D-finies, le fait de chercher exclusivement des majorants hyperexponentiels (solutions d'équations différentielles du premier ordre à coefficients polynomiaux) représente une limitation plus importante. ♢

L'algorithme 5.22 récapitule la procédure complète. Des variantes sont possibles suivant les possibilités du langage d'implémentation ou pour s'accommoder d'un corps de coefficients autre que les rationnels.

Algorithme 5.22. BoundRatpoly(r, P_α, m)

Entrée. Une fraction rationnelle irréductible $r = N/D \in \mathbb{Q}[z]$. Un polynôme $P_\alpha \in \mathbb{Q}[z]$ et un entier $m \geqslant 1$ tels que

$$\big(0 < \delta(P_\alpha) < \delta(D)\big) \vee \big(\delta(P_\alpha) = \delta(D)\big) \wedge \big(m \geqslant \nu_\delta(D)\big).$$

Sortie. Un rationnel $M \geqslant 0$ tel que $r(z) \trianglelefteqslant M\,(1 - z/\delta(P_\alpha))^{-m}$.

1 calculer le quotient $A \in \mathbb{Q}[z]$ et le reste $B \in \mathbb{Q}[z]$ de la division euclidienne de N par D *[on a $r = A + B/D$]*

2 calculer la décomposition sans carré $D = D_1\,D_2^2 \cdots D_k^k$

3 calculer les coefficients $h_{i,d} \in \mathbb{Q}[\zeta]$ de la décomposition en éléments simples de B/D écrite sous la forme $\dfrac{B(z)}{D(z)} = \displaystyle\sum_{i=1}^{k} \sum_{D_i(\zeta)=0} \sum_{d=1}^{i} \frac{h_{i,d}(\zeta)}{(\zeta - z)^d}$ (voir [42, §2.7])

4 pour i de 1 à k

  5 pour d de 1 à i

    6 calculer une borne $c_{i,d} \geqslant \displaystyle\sum_{D_i(\zeta)=0} |h_{i,d}(\zeta)\,\zeta^{-d}|$.

7 choisir $N_0 \geqslant \max\left(1, 1 + \deg A, \max_{i=m+1}^{k} \dfrac{i-m}{\log\left(\delta(D_i)/\delta(P_\alpha)\right)}\right)$

8 *[borne sur le reste]* poser

$$t(n) = \sum_{i=1}^{k} \sum_{d=0}^{i-1} c_{i,d}\, \frac{(n+1)^{\uparrow(d-1)}}{(n+1)^{\uparrow(m-1)}} \left(\frac{\delta(P_\alpha)}{\delta(D_i)}\right)^n$$

9 *[borne sur la tête du développement]* calculer par récurrence les coefficients de la série tronquée $r_{;N_0}(z) = \sum_{n=0}^{N_0-1} r_n\,z^n$, et choisir

$$h(N_0) \geqslant \max_{n=0}^{N_0-1} \left(\frac{|r_n|}{\binom{n+m-1}{m-1}\,\delta(P_\alpha)^n}\right)$$

10 renvoyer une approximation par excès de $\max\left(h(N_0), t(N_0)\right)$ ♢

En pratique, il est souhaitable que non seulement α et m mais aussi M soient aussi fins que possible. Pour améliorer la borne calculée, on peut répéter l'étape 9 de l'algorithme en augmentant N_0 jusqu'à ce que celui-ci ou la différence $t(N_0) - h(N_0)$ atteigne une limite fixée à l'avance. Une seconde possibilité est de renvoyer une borne avec partie polynomiale, de la forme

$$b(z) = \frac{M}{(1 - \alpha\,z)^m} + P(z), \qquad P(z) \in \mathbb{Q}[z],$$

ce qui autorise à choisir M plus petit. Dans l'algorithme 5.22, on prend par exemple

$$M = t(N_0), \qquad P(z) = \sum_{n=0}^{N_0} \max\left(0, |r_n| - M\,(n+1)^{\uparrow(m-1)}\,\alpha^n\right)$$

et l'on omet la ligne 9.

5.3.3 Séries majorantes pour les fonctions D-finies normalisées

Une fois calculées des bornes sur ses coefficients, nous en venons à l'application de la méthode de Cauchy-Kovalevskaya proprement dite pour majorer les solutions de l'équation différentielle (5.3) obtenue à l'issue de la normalisation de la section 5.2. Cette normalisation nous a ramenés au cas d'une équation différentielle avec un point régulier à l'origine.

La série majorante que nous obtenons est plus simple que $u(z)$ dans la mesure où elle est solution d'une équation différentielle du premier ordre.

REMARQUE 5.23. Les séries dont les coefficients satisfont des *récurrences* d'ordre 1, autrement dit, les séries hypergéométriques généralisées ${}_pF_q$, ne constituent pas une classe de majorants fins acceptable. En effet, la partie sous-exponentielle du comportement asymptotique de leurs coefficients est toujours polynomiale, interdisant par exemple les situations du type $u_n \sim e^{\sqrt{n}}$. ♢

Le procédé est le suivant. En isolant le coefficient constant du développement de Taylor en zéro de chaque coefficient, l'équation (5.3) se réécrit

$$Q(\theta)\cdot u = z\left(\tilde{a}^{[r-1]}\,\theta^{r-1} + \cdots + \tilde{a}^{[1]}\,\theta + \tilde{a}^{[0]}\right)\cdot u, \tag{5.11}$$

où $Q \in \mathbb{Q}[X]$ est un polynôme unitaire de degré r (le polynôme indiciel) et les $\tilde{a}^{[k]}$ sont des fractions rationnelles en z. On note m_k la multiplicité maximale d'un point du cercle $|z| = \delta(P_\alpha)$ comme pôle d'une des fractions $\tilde{a}^{[k]}$, et l'on pose

$$T = \max\Big(0, \max_{k=0}^{r-1}(m_k - r + k)\Big).$$

Soulignons que comme dans le cas des fractions rationnelles, si l'algorithme 5.26 prend P_α en entrée, l'objet essentiel de la méthode est de traiter le cas où $\delta(P_\alpha)$ est égal au module des singularités dominantes de l'équation (5.11). L'entier T est alors parfois appelé irrégularité (de Malgrange) de ces singularités, et d'après le critère de Fuchs, on a $T = 0$ si ce sont toutes des points singuliers réguliers.

La réduction de l'ordre de l'équation différentielle de r pour (5.11) à 1 pour ce qui fait office d'équation majorante repose sur le résultat suivant.

LEMME 5.24. Soit $b \in \mathbb{Q}[[z]]$ une série telle que

$$\forall n,\quad \forall j \leqslant n-1,\quad \left|\sum_{k=0}^{r-1} \tilde{a}^{[k]}_{n-1-j}\, j^k\right| \leqslant n^{r-1}\, b_{n-1-j}. \tag{5.12}$$

Soient $\lambda > 1$ et $N \geqslant 1$ tels que $n \geqslant N \Rightarrow n^r \leqslant \lambda\, Q(n)$. Si u et v sont des solutions respectivement de (5.11) et de

$$v'(z) = \lambda\, b(z)\, v(z) \tag{5.13}$$

avec $u_{;N} \trianglelefteqslant v_{;N}$, alors $u \trianglelefteqslant v$. ♢

DÉMONSTRATION. En extrayant le coefficient de z^n dans les équations (5.11) et (5.13), il vient pour tout $n \in \mathbb{N}$

$$Q(n)\, u_n = \sum_{j=0}^{n-1}\sum_{k=0}^{r-1} \tilde{a}^{[k]}_{n-1-j}\, j^k\, u_j \qquad n\, v_n = \lambda \sum_{j=0}^{n-1} b_{n-1-j}\, v_j.$$

Soit $n \geqslant N$ (on a alors $Q(n) > 0$), et supposons $|u_j| \leqslant v_j$ pour tout $j \leqslant n-1$. On a

$$Q(n)\,|u_n| \leqslant n^{r-1}\sum_{j=0}^{n-1} b_{n-1-j}\,|u_j| \leqslant \lambda\,\frac{Q(n)}{n}\sum_{j=0}^{n-1} b_{n-1-j}\, v_j = Q(n)\, v_n$$

d'où $u \trianglelefteqslant v$ par récurrence sur n. □

Le lemme 5.25 constitue l'étape essentielle de la majoration. L'énoncé est un peu plus général que nécessaire, en vue d'une application future.

LEMME 5.25. Supposons que les coefficients de (5.11) vérifient

$$\forall k,\quad \tilde{a}^{[k]} \trianglelefteqslant \binom{r-1}{k}\frac{M}{(1-\alpha\, z)^{r-k+T}},$$

et posons

$$b(z) = \frac{M}{(1-\alpha\, z)^{T+1}}.$$

On a alors

$$\forall \gamma \in \mathbb{C}, \quad \forall n \in \mathbb{N}, \quad \forall j \leqslant n-1, \quad \sum_{k=0}^{r-1} |\tilde{a}^{[k]}_{n-1-j}|\,|\gamma+j|^k \leqslant (|\gamma|+n)^{r-1}\, b_{n-1-j}.$$

En particulier, ce choix de b satisfait l'inégalité (5.12). ♢

Démonstration. Pour tout $k \in [\![0, r-1]\!]$, on a

$$\binom{n-1-j+T}{T}^{-1}\binom{n-1-j\ +\ r-k+T-1}{r-k+T-1} = \frac{(n-j+T)^{\uparrow(r-1-k)}}{(T+1)^{\uparrow(r-1-k)}} \leqslant (n-j)^{r-1-k},$$

et par conséquent

$$\begin{aligned}
&\frac{1}{b_{n-1-j}} \sum_{k=0}^{r-1} |\tilde{a}^{[k]}_{n-1-j}|\,|\gamma+j|^k \\
&\leqslant \binom{n-1-j+T}{T}^{-1} \sum_{k=0}^{r-1} \binom{r-1}{k}\binom{n-1-j+r-k+T-1}{r-k+T-1} |\gamma+j|^k \\
&\leqslant \sum_{k=0}^{r-1} \binom{r-1}{k} |\gamma+j|^k\,(n-j)^{r-1-k} \\
&= (|\gamma+j|+n-j)^{r-1} \leqslant (|\gamma|+n)^{r-1},
\end{aligned}$$

d'où le résultat. □

On utilise l'algorithme de la section précédente pour calculer des bornes de la forme

$$\tilde{a}^{[k]} \trianglelefteqslant \frac{M^{[k]}}{(1-\alpha\, z)^{r-k+T}} \tag{5.14}$$

où comme d'habitude $\alpha = \delta(P_\alpha)^{-1}$. On pose

$$M = \max_{k=0}^{r} \frac{M^{[k]}}{\binom{r-1}{k}};$$

et l'on choisit $K > M/\alpha$ puis $N \in \mathbb{N}$ tel que $M\, n^r \leqslant \alpha\, K\, Q(n)$ pour $n \geqslant N$. D'après les lemmes 5.25 et 5.24 (appliqués avec $b(z) = \alpha\, K\, (1-\alpha\, z)^{-T-1}$ et $\lambda = \alpha\, K/M$), toute solution u de (5.11) est majorée par une solution v de

$$v'(z) = \frac{\alpha\, K}{(1-\alpha\, z)^{T+1}}\, v(z), \tag{5.15}$$

soit

$$u(z) \trianglelefteqslant v(z) = \begin{cases} \dfrac{A}{(1-\alpha\, z)^K} & \text{si } T = 0 \\ A \exp \dfrac{K/T}{(1-\alpha\, z)^T} & \text{sinon.} \end{cases} \tag{5.16}$$

On ajuste enfin la constante d'intégration A pour satisfaire les inégalités initiales $v_n \geqslant |u_n|$ pour $n \leqslant N$. La proposition ci-dessous résume les conclusions de cette section.

Proposition 5.27. Soit $D \in \mathbb{Q}[z]\langle\theta\rangle$ un opérateur différentiel, dont on suppose qu'il admet un point régulier (éventuellement singulier) à l'origine, et que ses singularités dominantes sont finies et de module supérieur ou égal à ρ. Soit u une solution de l'équation $D \cdot u = 0$ analytique à l'origine. Étant donnés D, un polynôme P_α tel que $\rho = \alpha^{-1} = \delta(P_\alpha)$ et une procédure calculant le développement en série de u à l'origine tronqué à un ordre quelconque, la fonction BoundNormalDiffeq (algorithme 5.26)

ALGORITHME 5.26. BoundNormalDiffeq(D, P_α, u)

Entrée. Un opérateur différentiel $D = \sum_{k=0}^{r} a^{[k]}\,\theta^k \in \mathbb{Q}[z]\langle\theta\rangle$. Un polynôme $P_\alpha \in \mathbb{Q}[z]$ avec $\alpha^{-1} = \delta(P_\alpha)$. Une série $u \in \mathbb{Q}[[z]]$ calculable à un ordre arbitraire.

Sortie. Des paramètres de borne $T \in \mathbb{N}$, $K \in \mathbb{N}^*$ et $A > 0$.

1 pour k de 0 à $r-1$

  2 $c^{[k]} := \frac{a^{[k]}}{a^{[r]}}\Big|_{z=0}$ (ou échouer avec l'erreur « l'origine doit être un point régulier » en cas de division par zéro)

  3 $\tilde{a}^{[k]} := \frac{1}{z}\left(\frac{a^{[k]}}{a^{[r]}} - c^{[k]}\right)$

4 $T := \max\left(\{0\} \cup \left\{\nu_\delta\big(\operatorname{den}\tilde{a}^{[k]}\big) - r + k : 0 \leqslant k < r \text{ et } \delta\big(\operatorname{den}\tilde{a}^{[k]}\big) = \delta(P_\alpha)\right\}\right)$

5 pour k de 0 à $r-1$

  6 $M^{[k]} := \text{BoundRatpoly}\big(\tilde{a}^{[k]}, P_\alpha, T + r - k\big)$ (algorithme 5.22)
  $\lfloor ainsi\ \tilde{a}^{[k]} \preccurlyeq M^{[k]}\,(1-\alpha\,z)^{-T-r+k} \rfloor$

7 $M := \max_{k=0}^{r-1} \dfrac{M^{[k]}}{\binom{r-1}{k}}$

8 choisir $K \in \mathbb{N}^*$ tel que $K \geqslant 2\,M\,\delta(P_\alpha)$

9 $N \leftarrow 1$

10 tant que $\sum_{k=0}^{r-1} \left|c^{[k]}\right| N^k \geqslant \left(1 - \frac{M}{K}\,\delta(P_\alpha)\right) N^r$

  11 $N \leftarrow 2\,N$

12 calculer les coefficients $u_0, u_1, ..., u_{N+1}$ et $v_0, v_1, ..., v_{N+1}$ de u et v, où v est définie par (5.16) avec $A = 1$

13 $A := \max_{n=0}^{N} \dfrac{|u_n|}{v_n}$

14 renvoyer (T, K, A) ♢

calcule des paramètres $T \in \mathbb{N}$, $K \in \mathbb{N}^*$ et $A \in \mathbb{Q}_+$ tels que l'inégalité (5.16) soit vérifiée. ♢

En plus de son module ρ, la majoration préserve l'irrégularité T de la singularité dominante de l'équation différentielle donnée en entrée. L'irrégularité est liée à la croissance sous-exponentielle de la suite des coefficients.

Naturellement, l'algorithme s'applique non seulement aux fonctions D-finies mais aussi aux solutions d'équations différentielles dont les coefficients sont *majorés* par des séries rationnelles. On pourrait imaginer d'itérer le processus de majoration pour borner des fonctions plus compliquées que les fonctions D-finies.

5.3.4 Variante : majorants avec partie polynomiale

Nous avons noté que l'on pouvait améliorer les numérateurs $M^{[k]}$ des bornes sur les fractions rationnelles — et donc le paramètre K des majorants (5.16) — en ajoutant à celles-ci une partie polynomiale. Afin d'exploiter ces majorants, on remplace dans le raisonnement de la section précédente le lemme 5.25 par le suivant.

LEMME 5.28. Si les coefficients de (5.11) vérifient

$$\forall k, \quad \tilde{a}^{[k]} \preccurlyeq \binom{r-1}{k}\left(\frac{M}{(1-\alpha\,z)^{r-k+T}} + \sum_{n=0}^{d} (n+1)^{r-k} P_n\,z^n\right),$$

alors le choix de

$$b(z) = \frac{M}{(1-\alpha\,z)^{T+1}} + P(z), \qquad P(z) = \sum_{n=0}^{d} P_n\,z^n$$

satisfait l'inégalité (5.12). ♢

Démonstration. La preuve est pratiquement la même que celle du lemme 5.25 : en notant $f_n^{[k]} = [z^n]\,(1-\alpha\,z)^{-r+k-T}$, on a pour $0 \leqslant j \leqslant n-1$

$$\left|\sum_{k=0}^{r-1} \tilde{a}^{[k]}_{n-1-j}\, j^k\right| \leqslant M \sum_{k=0}^{r-1} \binom{r-1}{k} f^{[k]}_{n-1-j}\, j^k + P_{n-1-j} \sum_{k=0}^{r-1} \binom{r-1}{k} (n-j)^{r-k}\, j^k$$
$$\leqslant M\, f^{[r-1]}_{n-1-j}\, n^{r-1} + P_{n-1-j}\, n^{r-1}.$$

Le premier terme est exactement celui du lemme 5.25 avec $\gamma = 0$; le second provient de la définition de P. □

Les majorants $v(z)$ de l'équation (5.16) se trouvent donc multipliés par $\exp(\int P)$. Cela les fait malheureusement sortir de la classe que nous considérerons par la suite. Les estimations asymptotiques et les techniques pour en extraire les différentes bornes s'appliquent tout de même, mais les formes closes deviennent franchement peu lisibles, voire pour certaines ne sont plus disponibles. L'observation qui suit permet de remplacer le facteur exponentiel par une constante.

Proposition 5.29. Soit $v(z)$ défini par l'équation (5.16). Si f est une fonction entière à coefficients positifs, on a $f(z)\,v(z) \preccurlyeq f(\alpha^{-1})\,v(z)$. ♢

Démonstration. Supposons $\alpha = 1$. La suite (v_n) est alors croissante. (Pour $T > 0$, on peut le vérifier en écrivant

$$v_n = \sum_{k=0}^{\infty} \frac{1}{k!} \binom{n+kT-1}{n} \left(\frac{K}{T}\right)^n$$

et en formant la différence $v_{n+1} - v_n$.) On a donc

$$f(z)\,v(z) = \sum_{n=0}^{\infty} \sum_{i+j=n} f_i\, v_j\, z^n \preccurlyeq \sum_{n=0}^{\infty} \sum_{i=0}^{\infty} f_i\, v_n\, z^n = f(1)\,v(z).$$

Il s'ensuit pour α quelconque que $f(\alpha^{-1}\,z)\;v(\alpha^{-1}\,z) \preccurlyeq f(\alpha^{-1})\;v(\alpha^{-1}\,z)$, d'où le résultat en composant à droite avec $z \mapsto \alpha\,z$. □

5.3.5 Variante : points ordinaires

En un point *ordinaire*, la méthode de majoration directe exposée en §5.3.1 permet déjà d'obtenir des séries majorantes fines de la forme (5.16). Considérons une équation de la forme

$$y^{(r)}(z) + a^{[r-1]}(z)\,y^{(r-1)} + \cdots + a_0(z) = 0, \qquad a^{[k]} \in \mathbb{Q}(z), \tag{5.17}$$

où les $a^{[k]}$ n'ont pas de pôle en zéro. À l'aide de l'algorithme 5.22, on peut calculer M et N tels que

$$v^{(r)} = \frac{M}{(1-\alpha z)^N} \sum_{k=0}^{r-1} \binom{r-1}{k} N^{\uparrow r-k} \left(\frac{\alpha}{1-\alpha z}\right)^{r-k} v^{(k)}$$

soit une équation majorante de (5.17), ce qui donne une série majorante de la forme $v(z) = \exp\,(M\,(1-\alpha\,z)^{-N})$. Lorsqu'en outre la singularité dominante de (5.17) est régulière, on peut prendre comme majorant l'équation d'Euler

$$v^{(r)} = \sum_{k=0}^{r-1} \frac{M^{[k]}}{(1-\alpha z)^{r-k}}\, v^{(k)},$$

et alors $v(z) = A\,(1-\alpha\,z)^{-\lambda}$ convient, où λ est l'unique solution réelle strictement positive de

$$\alpha^r\,\lambda^{\uparrow r} - M^{[r-1]}\,\alpha^{r-1}\,\lambda^{\uparrow(r-1)} - \cdots - M^{[0]} = 0.$$

À nouveau, des $M^{[k]}$ convenables sont déterminés par l'algorithme 5.22. Les détails se trouvent dans mon rapport de stage de M2 [173, chap. 5].

5.4 Bornes sur les suites P-récursives générales

5.4.1 Algorithme

À ce stade, nous sommes en mesure de borner (u_n) par une suite (v_n) donnée par sa série génératrice $v(z) = \mathcal{L}_{p,q}\tilde{v}(z)$, où $\tilde{v}$ est une série explicite solution d'une équation différentielle linéaire du premier ordre, et

$$\mathcal{L}_{p,q} f = \sum_{n=0}^{\infty} \frac{f_n}{\psi_n} z^n.$$

Lorsque $p < 0$, on dispose aussi d'une expression intégrale de $\mathcal{L}_{p,q} f$ en fonction de f.

ALGORITHME 5.30. BoundRec$(R, [u_0, ..., u_{s-1}])$

Entrée. Un opérateur de récurrence $R = \sum_{k=0}^{s} b^{[k]} S^k \in \mathbb{Q}[n]\langle S\rangle$. Des conditions initiales $u_0, ..., u_{s-1} \in \mathbb{Q}$.

Sortie. Des paramètres de borne $\kappa \in \mathbb{Q}$, $T \in \mathbb{N}$, $P_\alpha \in \mathbb{Q}[z]$, $K \in \mathbb{R}_+$, $A \in \mathbb{R}_+$.

1 se ramener si nécessaire au cas $b^{[0]} \neq 0$ en remplaçant R par $R\,S^{-m}$ où $m = \min\{k \,|\, b^{[k]} \neq 0\}$

2 $(\kappa, P_\alpha) := \text{Asympt}(R)$ (algorithme 5.8)

- *[normaliser la récurrence et encoder la partie sous-exponentielle des solutions par une équation différentielle]*

3 $D := \text{Normalize}(R, \kappa)$ (algorithme 5.13)

- *[borner les solutions de cette équation différentielle]*

4 définir une procédure $\tilde{u}_{;\cdot}$ qui, étant donné $n \in \mathbb{N}$, calcule

$$\tilde{u}_{;n} \trianglerighteq \sum_{k=0}^{n-1} q^{-pk/q}\,\Gamma\!\left(\frac{k}{q}+1\right)^{-p} u_k\, z^k \qquad \left(\text{où } \frac{p}{q} = \kappa\right)$$

en déroulant la récurrence $R \cdot u = 0$ à partir des conditions initiales $u_0, ..., u_{s-1}$
[en pratique, il peut être plus commode de reporter ici le calcul du paramètre A]

5 $(T, K, A) := \text{BoundNormalDiffeq}(D, P_\alpha, \tilde{u}_{;\cdot})$ (algorithme 5.26)

6 renvoyer $(\kappa, P_\alpha, T, K, A)$ ♢

PROPOSITION 5.31. Soit $R \in \mathbb{Q}[n]\langle S\rangle$ un opérateur réversible non singulier d'ordre s. Soit $u \in \mathbb{Q}^{\mathbb{N}}$ une solution de la récurrence $R \cdot u = 0$. Étant donnés R et les conditions initiales $u_0, ..., u_{s-1} \in \mathbb{Q}$, l'algorithme 5.30 calcule $p/q = \kappa$, $\alpha = 1/\delta(P_\alpha)$, T, K et A tels que

$$\forall n \in \mathbb{N}, \qquad |u_n| \leqslant v_n = q^{\frac{p}{q}n}\,\Gamma\!\left(\frac{n}{q}+1\right)^{p} \tilde{v}_n \tag{5.18}$$

où $\tilde{v}$ est définie comme dans l'équation (5.16). On a en outre

$$\limsup_{n\to\infty} \left|\frac{u_n}{v_n}\right|^{1/n} = 1$$

pour un choix générique de $(u_0, ..., u_{s-1})$. ♢

DÉMONSTRATION. C'est une conséquence des propositions 5.8, 5.14 et 5.27. À l'issue de la ligne 1 de l'algorithme 5.30, l'opérateur R satisfait les hypothèses de la proposition 5.14. Par conséquent, l'opérateur D calculé ligne 3 annule $\tilde{u}_n = \sum_{n=0}^{\infty} \psi_n\, u_n\, z^n$, et la procédure définie à la ligne suivante calcule des approximations par excès de

troncatures de $\tilde{u}$. (Rappelons que nous avons adopté $\psi_n = q^{-pn/q}\,\Gamma(n/q+1)^{-p}$.) Il s'ensuit que $\tilde{u} \preccurlyeq \tilde{v}$ par la proposition 5.27, et donc que $u = \mathcal{L}_{p,q}\,\tilde{u} \preccurlyeq \mathcal{L}_{p,q}\,\tilde{v} = v$. Enfin, pour des conditions initiales génériques, on a

$$\operatorname*{limsup}_{n\to\infty} \left|\frac{u_n}{v_n}\right|^{1/n} = \operatorname*{limsup}_{n\to\infty} \left|\frac{u_n}{n!^\kappa\,\alpha^{n+o(1)}\,n^{O(1)}}\right|^{1/n} = 1$$

d'après la proposition 5.8. □

5.4.2 Formules explicites

La représentation (5.18) suffit dans certains contextes, mais des expressions plus explicites pour les coefficients v_n n'en sont pas moins désirables. Dans le cas $T = 0$, c'est-à-dire quand les singularités dominantes de l'équation différentielle qui annule la série génératrice obtenue après normalisation de $u(z)$ sont des points singuliers réguliers, le résultat est immédiat.

Proposition 5.32. Pour $T = 0$, le membre droit de la borne (5.18) est donné par

$$v_n = A\,\Gamma\!\left(\frac{n}{q}+1\right)^p \left(p^{\frac{p}{q}}\,\alpha\right)^n \binom{n+K-1}{K-1}. \qquad \diamondsuit$$

Démonstration. On a $\tilde{v}(z) = A\,(1-\alpha\,z)^{-K} = A\sum_{n=0}^{\infty}\binom{n+K-1}{K-1}\alpha^n\,z^n$. □

En présence d'une singularité dominante irrégulière, le coefficient général $\tilde{v}_n$ admet à nouveau une expression « en forme close » en termes de fonctions hypergéométriques généralisées.

Proposition 5.33. Lorsque $T > 0$, la suite $\tilde{v}_n$ qui apparaît dans la borne (5.18) est donnée par

$$\tilde{v}_n \;=\; A\,\alpha^n\,{}_T\mathrm{F}_T\left(\begin{array}{cccc} \frac{n+T}{T} & \frac{n+T+1}{T} & \cdots & \frac{n+2T-1}{T} \\ \frac{T+1}{T} & \frac{T+2}{T} & \cdots & \frac{2T}{T} \end{array}\,\middle|\, \frac{K}{T}\right). \qquad \diamondsuit$$

Démonstration. On calcule :

$$\begin{aligned} \exp\frac{K/T}{(1-\alpha\,z)^T} &= \sum_{k=0}^{\infty}\frac{1}{k!}\left(\frac{K}{T}\right)^k \left(\sum_{j=0}^{\infty}\binom{j+T-1}{T-1}\alpha^\ell\,z^\ell\right)^k \\ &= \sum_{n=0}^{\infty}\sum_{k=0}^{\infty}\frac{1}{k!}\left(\frac{K}{T}\right)^k \sum_{j_1+\cdots+j_k=n}\prod_{i=1}^{k}\binom{j_i+T-1}{T-1}\alpha^n\,z^n \\ &= \sum_{n=0}^{\infty}\sum_{k=0}^{\infty}\frac{1}{k!}\binom{n+Tk-1}{Tk-1}\left(\frac{K}{T}\right)^k \alpha^n\,z^n. \end{aligned}$$

La dernière égalité s'obtient par exemple en comptant de deux façons différentes les compositions de n en Tk sommants. □

Mais on peut aussi majorer $\tilde{v}_n$ lui-même par une expression plus simple sans abandonner la finesse asymptotique de la borne à l'aide d'une version simple de la méthode du col. Puisque $\tilde{v}$ est à coefficients positifs, on a

$$\forall n \in \mathbb{N}, \quad \forall t \in [0; 1/\alpha[, \quad \tilde{v}_n \leqslant \frac{\tilde{v}(t)}{t^n} \tag{5.19}$$

(soit, en termes de séries majorantes, $\tilde{v}(z) \leqslant \tilde{v}(t)/(1-{}^z\!/_t)$).

La dérivée logarithmique de v est $v'(z)/v(z) = \alpha\, K/(1-\alpha\, z)^{T+1}$ d'après (5.15). À n fixé, le membre droit de (5.19) est donc minimal pour l'unique $t = t_n \in [0;1[$ tel que $K\,\alpha\, t_n = n\,(1-\alpha\, t_n)^{T+1}$; et celui-ci satisfait

$$1-\alpha\, t_n \sim \left(\frac{K}{n}\right)^{\frac{1}{T+1}} \qquad \text{quand } n\to\infty.$$

Cette approximation nous convient bien : posons

$$r_n = \frac{1}{\alpha}\left(1-\left(\frac{K}{n+K+1}\right)^{\frac{1}{T+1}}\right) \tag{5.20}$$

(le terme $K+1$ au dénominateur ne change pas le comportement asymptotique mais assure que $0<r_n<\alpha^{-1}$). L'inégalité (5.19) fournit les majorations suivantes.

PROPOSITION 5.34. Pour $T>0$, la suite $(\tilde v_n)$ satisfait

$$\tilde v_n \leqslant A\,\frac{\tilde v(r_n)}{r_n^n} = \alpha^n\,\frac{\exp\left(\frac{K}{T}\left(\frac{n+K+1}{K}\right)^{\frac{T}{T+1}}\right)}{\left(1-\left(\frac{K}{n+K+1}\right)^{\frac{1}{T+1}}\right)^n} = O\Big(\alpha^{n+O(n^{T/(T+1)})}\Big) \tag{5.21}$$

où r_n est défini par (5.20). De même, on a

$$\tilde v_n \leqslant A\,\alpha^n\left(\frac{n+K+1}{K}\right)^K\left(1-\frac{K}{n+K+1}\right)^{-n} = O\big(\alpha^n\, n^{O(1)}\big) \tag{5.22}$$

pour $T=0$. □

Pour en revenir à v_n, on peut écrire une majoration de la forme $v_n \leqslant A\, n!^\kappa\, \alpha^n\, \varphi(n)$ où $\varphi(n)$ est complètement explicite, en combinant les bornes sur $\tilde v_n$ découlant des propositions 5.32 à 5.34 avec la relation entre ψ_n et $n!$ explicitée dans le lemme suivant. On a alors

$$|u_n| \leqslant v_n \leqslant A\, n!^\kappa\, \alpha^n\, \varphi(n)$$

et $\log(\varphi(n)) = o(n)$. Ceci conclut la preuve du théorème 5.2.

LEMME 5.35. Pour $p\in\mathbb{Z}$, $q\in\mathbb{N}^*$ et $n\geqslant 3\,q/2$, on a

$$\frac{1}{\psi_n} = q^{-\frac{p}{q}n}\,\Gamma(n/q+1)^p \leqslant \begin{cases} (2\,\pi)^{p/q}\,(n/q+1)^p\, n!^{p/q}, & p>0 \\ n^{-p/q}\, n!^{p/q}, & p<0. \end{cases} \qquad \diamondsuit$$

DÉMONSTRATION. La fonction Γ est croissante sur $[\frac{3}{2},\infty[$, donc

$$\prod_{k=0}^{q-1}\Gamma(n/q+k/q) \leqslant \Gamma(n/q+1)^q \leqslant \prod_{k=0}^{q-1}\Gamma(n/q+k/q+1).$$

D'après le théorème de multiplication [7, formule 6.1.20]

$$\Gamma(qz) = (2\pi)^{(1-q)/2}\, q^{qz-1/2}\prod_{k=0}^{q-1}\Gamma\Big(z+\frac{k}{q}\Big) \qquad (z\in\mathbb{C}),$$

appliqué à $z=n/q$, on en déduit

$$\frac{(2\,\pi)^{(q-1)/2}}{n\,q^{-1/2}} \leqslant \frac{q^n\,\Gamma(n/q+1)^q}{\Gamma(n+1)} \leqslant \frac{(2\,\pi)^{(q-1)/2}\,(n+1)^{\uparrow(q-1)}}{q^{q-1/2}}$$

d'où le résultat en élevant à la puissance p/q l'une ou l'autre des inégalités suivant le signe de p. □

5.5 Implémentation

Les algorithmes décrits ci-dessus sont implémentés dans NumGfun, parfois à de petites variations près (voir aussi §1.2.7 et figure 1.1). Le code de calcul de bornes est relativement indépendant du reste du module.

La fonction `bound_rec` correspond à la situation du théorème 5.1 : elle prend en entrée une récurrence linéaire à coefficients polynomiaux et des conditions initiales, et calcule une borne sur la solution, grâce à l'algorithme présenté en §5.4. De même, `bound_ratpoly` et `bound_diffeq` calculent des séries majorantes fines respectivement pour les fractions rationnelles (comme décrit en §5.3.2) et pour les fonctions D-finies (par l'algorithme de la section 5.3.3 dans le cas normalisé, en enrobant formellement le résultat de `bound_rec` sinon). Enfin — en anticipant sur le chapitre suivant — deux fonctions `bound_diffeq_tail` et `bound_rec_tail` servent à calculer des bornes sur les sommes et les restes de séries D-finies convergentes.

Tous les exemples de ce chapitre et du suivant ont été calculés à l'aide de cette implémentation. Elle est aussi utilisée dans le DDMF pour fournir des séries majorantes des développements de Taylor des fonctions décrites et pour calculer des ordres de troncature de ces séries assurant une certaine qualité d'approximation sur un disque, et sert de base au calcul automatique de bornes d'erreur requis par les fonctionnalités d'évaluation numérique de NumGfun.

5.6 Remarques finales

5.6.1 Bornes fines sans formule explicite

Il n'est pas indispensable de passer par des séries majorantes fines pour obtenir des bornes v_n telles que $|u_n| \leqslant v_n$ et $\limsup |u_n/v_n|^{1/n} \leqslant 1$ par la méthode de Cauchy-Kovalevskaya — du moins si l'on renonce à avoir une expression simple de v_n en fonction de n. L'idée est simplement de calculer des séries majorantes de rayon de convergence plus petit, comme dans l'exemple 5.19, mais *choisi en fonction de* n, et approchant celui de $u(z)$ à vitesse convenable quand n tend vers l'infini. L'avantage est que l'on peut se contenter de séries majorantes très simples, de la forme $A/(1-\nu z)$; le prix à payer, que l'on calcule essentiellement une équation majorante par valeur de v_n cherchée.

Considérons ainsi l'équation $D \cdot u = \sum_k a^{[k]}(z)\, u^{(k)}(z) = 0$, où l'on suppose que l'origine est un point ordinaire. Soit comme d'habitude α l'inverse du module de la singularité dominante de D, et soit $\nu > \alpha$. Van der Hoeven [237, §3.5] montre que l'on peut calculer un réel C indépendant de ν tel que

$$a^{[k]} \trianglelefteq \frac{M(\nu)}{1-\nu z}, \qquad k = 0, \ldots, r-1, \tag{5.23}$$

entraîne

$$u(z) \trianglelefteq \frac{C}{(1-\nu z)^{\lceil (M(\nu)+1)/\nu \rceil}}.$$

On suppose de plus que la méthode utilisée pour établir les inégalités (5.23) garantit que pour un certain d,

$$M(\nu) = O\left(\sup_{k \in \mathbb{N}} \left(k^d \left(\frac{\alpha}{\nu} \right)^k \right) \right) \quad \text{quand } \nu \to \alpha.$$

C'est le cas de l'algorithme 5.22, en prenant pour d le maximum des multiplicités m_k définies en début de §5.3.3. Mimant (5.20), on adopte (par exemple)

$$\nu = \nu_n = \left(1 + n^{-\frac{1}{2d}} \right) \alpha.$$

Comme

$$\sup_{k \in \mathbb{N}} \left(k^d \left(\frac{\alpha}{\nu} \right)^k \right) = O\left(\left(\log \frac{\nu}{\alpha} \right)^{-d} \right) \quad \text{quand } \nu \to \alpha,$$

il vient $M(\nu_n) = O(\sqrt{n})$, et

$$\begin{aligned} |u_n| &\leqslant C \binom{n + \lceil (M(\nu_n)+1)/\nu_n \rceil + 1}{n} \nu_n^n \\ &= O\Big(\binom{n + O(\sqrt{n})}{n} (1 + n^{-2d})^n \, \alpha^n \Big) \\ &= O\Big(\alpha^n \exp O\big(n^{1-1/(2d)} + \sqrt{n} \log n \big) \Big) \end{aligned}$$

Cela suggère de choisir $\nu = \big(1 + 1/n^{\Theta(1/d)}\big)$ dans les algorithmes de [236, 237].

5.6.2 Limitations

Compromis finesse-simplicité. Nous avons vu que des séries majorantes rationnelles ou hypergéométriques ne suffisent pas à atteindre le niveau de finesse qui nous intéresse ici. À l'inverse, il faudrait adopter une classe de majorants différente pour faire mieux. Le comportement en $\exp(\sqrt[3]{n})$ des coefficients de la série $\exp(1/\sqrt{1-z})$, par exemple, n'est pas représentable par nos majorants, qui ont nécessairement une partie sous-exponentielle de la forme $\exp\big(O(n^{1-1/T})\big)$.

De même, les majorants hyperexponentiels qui font l'affaire dans le cas d'un rayon de convergence fini ne recouvrent que des comportements en $n!^{1/k}\, e^{O(n)}$ dans celui des séries entières. Conserver des majorants fins qui s'écrivent comme composées de fonctions usuelles dans le cas le plus général demanderait d'introduire des briques de base moins communes (fonctions de Mittag-Leffler par exemple), d'où le choix de s'en tenir à des transformées de fonctions hyperexponentielles.

Singularités apparentes. Une équation différentielle peut avoir des points singuliers où aucune de ses solutions n'est singulière. De tels points singuliers sont appelés singularités apparentes. Par exemple, l'équation

$$(2\,z+1)\,y'''(z) + (4\,z^2-3)\,y''(z) + (-4\,z^2-2\,z+2)\,y'(z) = 0 \tag{5.24}$$

est singulière pour $z = -\frac{1}{2}$ alors que toutes ses solutions sont entières — l'espace de ses solutions est engendré par la fonction d'erreur, l'exponentielle et les fonctions constantes. La série majorante que calculent les algorithmes de ce chapitre pour les solutions de (5.24) a donc pour rayon de convergence $\frac{1}{2}$, ce qui en fait une borne excessivement mauvaise. Une façon de calculer néanmoins des bornes fines dans ce cas consiste à *désingulariser* l'opérateur. Par exemple, on peut montrer que toute solution y de (5.24) satisfait aussi l'équation

$$y^{(4)}(z) - 3\,y'''(z) - 4\,(z+2)\,(z-1)\,y''(z) + 2\,(2z^2+2\,z-3)\,y'(z) = 0$$

qui conduit quant à elle à la borne « en $n!^{-1/2}$ » attendue. Le processus de désingularisation est algorithmique [233, 63].

Conditions initiales non génériques. Les exemples que voici sont typiques des cas où notre méthode ne trouve pas de borne fine, en raison de conditions initiales « non génériques ».

EXEMPLE 5.36. Dans sa preuve de l'irrationalité de $\zeta(3)$, Apéry [8] introduit deux suites (a_n) et (b_n) solutions de

$$(n+2)^2\,u_{n+2} = (2\,n+3)\,(17\,n^2+51\,n+39)\,u_{n+1} - (n+1)^3 \tag{5.25}$$

avec

$$a_0 = 1, \quad a_1 = 5, \quad b_0 = 0, \quad b_1 = 6.$$

Appliqué à la récurrence (5.25) avec les conditions initiales $u_0 = -\zeta(3)$, $u_1 = 6 - 5\,\zeta(3)$, l'algorithme 5.30 détermine que

$$|b_n - \zeta(3)\,a_n| \leqslant 1{,}21\,(n^2 + 3\,n + 2)\,(17 + 12\,\sqrt{2})^n.$$

(On a $17 + 12\sqrt{2} \approx 33{,}97$.) Cette borne reflète le comportement asymptotique des suites (a_n) et (b_n). Mais l'argument d'Apéry est précisément que $b_n - \zeta(3)\,a_n$ tend rapidement vers 0 quand n tend vers l'infini, de sorte que les b_n/a_n sont de « trop bons » approximants rationnels de $\zeta(3)$ pour que celui-ci soit lui-même rationnel. ♢

Exemple 5.37. Comme nous le verrons au chapitre 9, les coefficients du développement de Tchebycheff sur $[-1, 1]$

$$u(z) = \frac{u_0}{2} + \sum_{n=1}^{\infty} c_n\,T_n(z)$$

d'une fonction D-finie donnée par une équation différentielle sans singularité dans

$$E_\rho = \{z \in \mathbb{C} : |z + \sqrt{z^2 - 1}| < \rho\} \qquad (\rho > 1)$$

forment une suite P-récursive et satisfont $c_n = O(\rho^{-n+o(n)})$. On pourrait songer à borner cette suite par les méthodes de ce chapitre de manière à en tirer une borne sur les restes de la série. Hélas, on peut montrer que la récurrence « minimale » en un certain sens [157, 21] qui annule la suite c_n admet aussi une solution c'_n telle que $\limsup_{n\to\infty} |c_n|^{1/n} \geqslant \rho$. Non seulement les bornes que l'on obtient par ce procédé ne sont pas fines, mais elles ne suffisent même pas à justifier la convergence de la série de Tchebycheff. ♢

Que faire ? Nous avons déjà relevé que calculer des bornes fines dans les situations non-génériques ne semblait pas faisable sans progrès majeur. Une réponse au problème suivant, s'il s'avérait plus facile, suffirait à contourner la difficulté dans un grand nombre de cas, dont celui des séries de Tchebycheff.

Question 5.38. Soient données une suite P-récursive u et deux réels κ, α avec la garantie que $|u_n| = O(n!^\kappa\,\alpha^{n+o(n)})$. Peut-on alors calculer une fonction φ telle que $|u_n| \leqslant n!^\kappa\,\alpha^n\,\varphi(n)$ pour tout n, et $\varphi(n) = e^{o(n)}$? ♢

Observons que si par exemple $u_n = w_n + O(\varepsilon_n)$ où w est une suite connue, elle-même P-récursive, et ε_n est suffisamment petite, cela permettrait de borner $|u_n - w_n|$ et donc d'obtenir des bornes inférieures sur $|u_n|$.

Chapitre 6

Restes de séries D-finies

« Les ρ se biffent. »

— Bruno Salvy

Le chapitre précédent a montré comment majorer automatiquement une suite donnée par une récurrence linéaire à coefficients polynomiaux qui la définit. Il s'agit maintenant d'en déduire des bornes sur les restes de séries D-finies, tant comme expressions lisibles par un humain (bornes « symboliques ») que pour calculer à partir d'une borne d'erreur sur la somme un ordre de troncature convenable (bornes « numériques »). Les sections 6.1.1 à 6.3.3 sont adaptées de [175], avec de petites extensions. Les suivantes développent des idées mentionnées sans plus de détails dans [174].

6.1 Problématique

6.1.1 Bornes symboliques sur les restes de séries

On se propose d'étendre les idées du chapitre précédent pour obtenir des bornes comme celles des exemples ci-dessous.

Exemple 6.1. Des algorithmes parmi les plus rapides pour calculer π à grande précision font appel à la formule

$$\sum_{k=0}^{\infty} t_k = \frac{640320^{3/2}}{12\,\pi} \qquad \text{où} \qquad t_k = \frac{(-1)^k\,(6\,k)!\,(13591409+545140134\,k)}{(3\,k)!\,(k!)^3\,640320^{3k}}, \tag{6.1}$$

due à Chudnovsky et Chudnovsky [56, p. 389]. En appliquant la méthode de la section 5.3 à la récurrence du premier ordre évidente sur les t_k, on obtient

$$\left|\sum_{k=n}^{\infty} t_k\right| \leqslant 10^6\,(2{,}3\,n^3 + 13{,}6\,n^2 + 25\,n + 13{,}6)\,\alpha^n$$

où $\alpha = {}^1\!/_{151931373056000} \approx 0{,}66 \cdot 10^{-14}$. Chaque terme de la série donne donc environ 14 chiffres décimaux corrects, et l'on déduit facilement de cette inégalité un ordre de troncature suffisant pour calculer π à toute précision donnée. ♢

Exemple 6.2. De même, à partir de l'équation différentielle

$$z\,\mathrm{Si}^{(3)}(z) + 2\,\mathrm{Si}''(z) + z\,\mathrm{Si}(z) = 0, \qquad \mathrm{Si}(0) = 0, \mathrm{Si}'(0) = 1,$$

le résultat de notre algorithme montre que la fonction sinus intégral est approchée sur le disque $|z| \leqslant 1$ avec une erreur absolue bornée uniformément par 10^{-100} par sa série de Taylor en zéro tronquée à l'ordre 74. Ce résultat est du même type que ceux affichés dans le DDMF (§2.2). ♢

Dans les exemples 6.1 et 6.2, on cherche en fait à majorer la suite $r_n = \sum_{k=n}^{\infty} t_k$, ou respectivement $r_n = u_n;(z)$ des restes d'une série dont le terme général est P-récursif.

NOTATION 6.3. Rappelons que si $u \in \mathbb{C}[[z]]$, on note $u_n = [z^n]\, u$ le coefficient de z^n dans u et $u_{n;}(z) = \sum_{k=n}^{\infty} u_k\, z^k$ le reste d'ordre n de u. On écrit aussi $u_{n;}^{(j)}$ pour $(u^{(j)})_{n;}$ (et non pas pour $(u_{n;})^{(j)}$). ♢

D'après les propriétés générales rappelées en §3.3 (proposition 3.15), ces suites de restes sont elles-mêmes P-récursives. Pourtant, on ne peut pas appliquer directement la borne du théorème 5.1. Faute de conditions initiales pour calculer la constante A, d'abord : au moins dans le premier des deux exemples, la valeur de r_0 est précisément ce que l'on cherche *in fine* ! Et surtout, quand bien même on saurait borner lesdites conditions initiales, le problème tomberait dans le cas non-générique du chapitre précédent, où une majoration des premiers termes n'est d'aucune aide.

En revanche, il n'est pas difficile de sommer pour $k \geqslant n$ les bornes sur les termes généraux données par l'algorithme de calcul de majorants. Plus généralement, si $u \preccurlyeq v$, les propriétés des séries majorantes (lemme 5.18) entraînent $|u_{n;}(\zeta)| \leqslant v_{n;}(|\zeta|)$. Le premier objectif de ce chapitre est de systématiser cette idée, en explicitant des bornes sur les restes $v_{n;}(x)$ des séries majorantes susceptibles d'être renvoyées par l'algorithme 5.30. En vue des applications au prolongement analytique numérique développées dans la suite, nous allons en fait considérer le problème légèrement plus général de borner les restes $|(u_{n;}^{(j)})(\zeta)|$ de dérivées de u en un point ζ de module $|\zeta| < \delta(a^{[r]})$, où $a^{[r]}$ désigne le coefficient de tête de l'équation différentielle comme solution de laquelle est donnée $u(z)$.

L'idée de base, liée à la méthode du col de l'analyse asymptotique (la même qui a servi à extraire certaines des bornes du précédent chapitre), est que des inégalités comme

$$g_{n;}(x) \leqslant \left(\frac{x}{t_n}\right)^n \frac{g(t_n)}{1 - x/t_n} \qquad (g \in \mathbb{R}_+[[z]], x \geqslant 0)$$

sont asymptotiquement fines quand $n \to \infty$ pour t_n convenablement choisi.

6.1.2 Application à l'évaluation numérique

L'application majeure des bornes sur restes de séries est l'approximation de fonctions, tout particulièrement, pour nous, en vue de leur évaluation numérique. Il s'agit de déterminer *automatiquement* des bornes d'erreur qui relient les approximations intervenant dans le calcul aux fonctions qu'elles représentent. Dans les sections 6.3.2 et 6.4, nous ferons quelques ajustements aux algorithmes de calcul de majorants en ciblant plus spécifiquement l'algorithme de prolongement analytique numérique du chapitre 8. Mais passons tout d'abord en revue quelques alternatives existantes.

Bornes *a priori*. Dans le cas qui nous intéresse, celui des séries entières tronquées, divers résultats élémentaires d'analyse réelle et complexe sont communément utilisés pour obtenir des bornes d'erreur. Citons le critère des séries alternées, la formule intégrale de Cauchy, et différentes variantes de la formule de Taylor avec reste intégral.

Karatsuba décrit ainsi des algorithmes d'évaluation à grande précision, bornes comprises, pour un certain nombre de fonctions spéciales dont la fonction hypergéométrique ${}_2\mathrm{F}_1$ (voir [138] et ses références). Du et Yap [79] donnent un algorithme de calcul de bornes sur les restes de la série hypergéométrique générale ${}_p\mathrm{F}_q$ évaluée en un point réel, *via* une analyse détaillée des variations de ses coefficients.

Dans un contexte différent, Neher [190] majore les valeurs atteintes sur un disque par une fonction analytique en se servant d'arithmétique d'intervalles complexe, et en tire des bornes sur les coefficients et les restes de son développement en série entière grâce à la formule de Cauchy. Cette méthode s'applique à toute fonction analytique « suffisamment explicite », notamment aux composées de fonctions usuelles. Elle est implémentée dans un logiciel appelé ACETAF[1] [82].

1. La qualité des bornes obtenues dépend fortement du choix judicieux de paramètres dont le réglage est laissé à l'utilisateur. Il pourrait être intéressant de combiner l'approche d'ACETAF avec

La méthode des séries majorantes, sur laquelle repose l'approche adoptée au chapitre précédent, est encore un outil classique. Avant nous, van der Hoeven l'a exploitée pour borner les restes de séries D-finies[2] au voisinage d'un point ordinaire de leur équation différentielle [236, §2.4], puis dans un contexte plus général qui couvre toute une gamme d'équations fonctionnelles [237]. L'algorithme présenté en §5.3.2 de ce mémoire peut être vu comme un raffinement de ceux de [237, § 3.5, §5.2]. Si l'on met de côté le contexte des *suites* P-récursives, la principale originalité de notre version est la finesse asymptotique des bornes obtenues.

Méthodes d'inclusion fonctionnelle. Quand on évalue une fonction ou que l'on en calcule une approximation, on n'est pas forcé de majorer les erreurs *a priori* pour garantir une certaine précision. Il est souvent plus facile de calculer des bornes d'erreur en parallèle d'approximations successives du résultat. On augmente alors progressivement la précision des calculs jusqu'à ce que la borne sur la qualité de l'approximation calculée soit plus petite que l'erreur cible.

Depuis les travaux précurseurs de Moore il y a un demi-siècle [181], l'encadrement numérique certifié de solutions d'équations différentielles — et d'équations fonctionnelles plus générales — fait l'objet d'une littérature abondante dans le domaine de l'arithmétique d'intervalles [179, 70, 209, 72, 180]. C'est une ligne de recherche qui rejoint certaines problématiques du calcul formel [240, 236], à travers l'emploi de représentations symboliques-numériques des fonctions, en général formées d'une approximation polynomiale à coefficients flottants accompagnée d'une borne d'erreur. Plusieurs structures de données de ce type ont été définies et étudiées indépendamment autour des années 1980. Les *modèles de Taylor* constituent la plus populaire de nos jours et la plus développée en pratique. Les panoramas de Makino et Berz [164] ainsi que de Neumaier [192] comparent les propriétés des modèles de Taylor à celles de différentes alternatives. Concernant plus particulièrement la résolution d'équations différentielles ordinaires à l'aide de modèles de Taylor, je renvoie le lecteur à Hoefkens [129] ou à Neher *et al.* [191].

Le plus intéressant pour nous est la manière dont on parvient à calculer des bornes d'erreur sur l'approximation de fonctions qui ne sont données au départ que par des équations implicites. L'idée commune est d'écrire le problème comme une équation de point fixe $u = \Phi(u)$ et de faire agir Φ sur des séries tronquées augmentées de bornes d'erreur. Si Φ est, disons, un opérateur intégral, on l'étend par des règles comme

$$\int^x (a_0 + a_1\,t + a_2\,t^2 + [\alpha,\beta])\,\mathrm{d}t \subseteq \int^x (a_0 + a_1\,t)\,\mathrm{d}t + B\!\left(a_3\,\frac{x^3}{3}\right) + [\alpha,\beta]\,B(x).$$

Dans cette formule, $B(p)$ est un intervalle, calculé à partir du domaine de variation de x, qui contient les valeurs possibles pour $p(x)$. Pour résoudre l'équation, on calcule une solution approchée sous forme de série entière tronquée

$$p(x) = a_0 + a_1\,x + \cdots + a_n\,x^n.$$

On recherche itérativement un intervalle $[\alpha,\beta]$ suffisamment petit tel que

$$\Phi(p + [\alpha,\beta]) \subseteq p + [\alpha,\beta], \tag{6.2}$$

en augmentant l'ordre de troncature n ou en réduisant le domaine où varie x autant que nécessaire. Sous des hypothèses peu contraignantes, l'existence d'une inclusion du type (6.2) implique celle d'une solution $u \in p + [\alpha,\,\beta]$ à l'équation $\Phi(u) = u$.

---

des outils d'analyse asymptotique du type analyse de singularité ou méthode du col dans l'espoir de calculer automatiquement des bornes fines.

2. Toujours dans le cas des fonctions D-finies, il avait auparavant proposé une méthode *ad hoc*, au fond guère différente de celle à base de séries majorantes [235, §2.2]. En un mot, si $B(n)$ est la matrice de la récurrence qui donne les coefficients de la fonction étudiée et P une matrice de changement de base (indépendante de n) convenable, il s'agit de borner l'écart entre $P^{-1}\,B(n)\,P$ et sa limite quand $n \to \infty$, qui est diagonale.

Si par exemple[3] Φ est un opérateur contractant sur un espace métrique complet dont $p+[\alpha,\beta]$ est un sous-espace fermé, le théorème du point fixe de Banach appliqué à la restriction de Φ à $p+[\alpha,\ \beta]$ montre que l'unique point fixe de Φ se trouve dans $p+[\alpha,\beta]$.

Limites des méthodes d'inclusion. Ce genre de technique est réputé fournir des encadrements fins à coût raisonnable pour des calculs à précision machine, y compris dans le cas d'équations non linéaires avec un grand nombre de variables. Deux difficultés les rendent insuffisantes vis-à-vis de nos objectifs.

Tout d'abord, les méthodes d'inclusion fonctionnelle présupposent habituellement qu'on dispose explicitement du développement en série d'une solution approchée pour lui appliquer l'opérateur de point fixe et vérifier une inclusion du type (6.2). Or, dans l'algorithme de prolongement analytique numérique de Chudnovsky et Chudnovsky (chapitre 8), il est essentiel d'éviter de calculer explicitement les polynômes d'approximation qui interviennent. On obtient directement leur valeur en un point grâce à une récurrence satisfaite par leurs coefficients. Cet obstacle ne semble pas infranchissable, et il serait intéressant de comprendre dans quelle mesure les méthodes d'inclusion s'adaptent à ce contexte. On peut espérer arriver à une méthode de certification du résultat moins lourde que celle de ce mémoire, et surtout plus facile à implémenter dans un langage de bas niveau.

Seconde limitation : il n'existe à ma connaissance pas de résultat de finesse pour les méthodes d'inclusion comparable à ceux que nous visons ici. Les conditions sous lesquelles celles-ci produisent des encadrements arbitrairement fins ne semblent pas complètement claires. À plus forte raison, il paraît délicat de s'en servir pour contrôler une fonction sur la totalité de son disque de convergence. Or, à nouveau dans l'application au prolongement analytique, la garantie de finesse des bornes se traduit dans la complexité de l'algorithme.

À l'inverse, les séries majorantes codent toutes les bornes nécessaires à l'algorithme du chapitre 8 en un seul objet relativement facile à contrôler analytiquement.

6.2 Restes de séries

Nous débutons par un petit catalogue de lemmes techniques destinés à déduire des séries majorantes obtenues au chapitre précédent différentes bornes utiles sur les restes des séries majorées. Ils prennent la forme d'inégalités générales, énoncées en fonction des paramètres κ, α, T, etc. renvoyés par l'algorithme 5.30. Cette généralité conduit parfois à des estimations pessimistes, et il est souvent possible de raffiner les inégalités au cas par cas.

6.2.1 Majoration fine de $u_n;(z)$ quand $n\to\infty$

Considérons l'équation différentielle

$$D\cdot y = a^{[r]}\,y^{(r)} + a^{[r-1]}\,y^{(r-1)} + \cdots + a^{[0]}\,y = 0, \tag{6.3}$$

avec $a^{[0]},\ldots,a^{[r]}\in\mathbb{Q}[z]$. Pour que la notion de bornes sur les restes ait un sens, nous nous plaçons dans le cas où l'origine est un point régulier de D, de sorte que toutes les solutions séries formelles sont convergentes. Ce peut être un point *singulier* régulier, et il peut alors y avoir des solutions qui ne s'expriment pas comme séries formelles. Nous nous occuperons d'elles plus loin.

3. Le livre de Kaucher et Miranker [139] présente une théorie générale de la résolution numérique certifiée d'équations dans les espaces fonctionnels, qui englobe différentes sortes de développements sur diverses bases de fonctions. Elle s'appuie sur des variantes de théorèmes de point fixe classiques, qui visent à valider par le calcul, non seulement l'inclusion, mais aussi les propriétés des opérateurs nécessaires à l'application des théorèmes, comme dans notre exemple le fait que Φ soit contractant. Nous aurons recours à une méthode de ce genre dans un cas simple au chapitre 9.

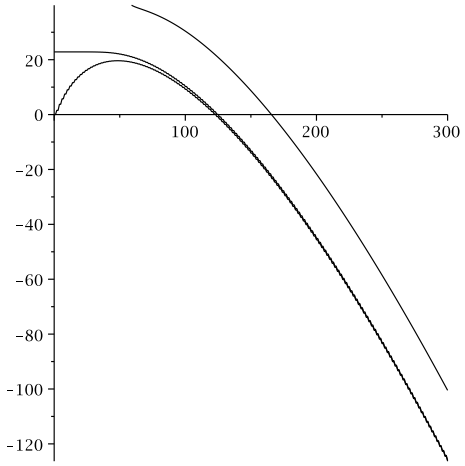


Figure 6.1. Restes du développement en série en zéro de la fonction d'erreur erf et de sa borne. De bas en haut, les valeurs en fonction de n de $\log(\mathrm{erf}_n;5)$, $\log|\mathrm{erf}_{n;}(5\,i)|$ et $\log(b_n)$, où b_n représente la borne (6.6) instanciée avec des paramètres calculés par l'algorithme 5.30. Le module du terme général de chacune de ces séries, de l'ordre de $5^n\,n!^{1/2}$, ne commence à décroître que lorsque $n \gtrsim e\,5^2 \approx 68$. On observe que la borne épouse la forme de la « bosse » qui résulte de ce phénomène, et se rapproche progressivement de la valeur exacte.

Comme pour le calcul de bornes sur les u_n, nous supposons calculés $\kappa = p/q$ et $\tilde{v} \in \mathbb{R}_+[[z]]$ tels que

$$u(z) \lhd v(z) = \mathcal{L}_{p,q} \cdot \tilde{v}(z) = \sum_{n=0}^{\infty} \psi_n\, \tilde{v}_n\, z^n,$$

avec $p \leqslant 0$ puisque nous nous limitons au cas convergent. La série $\tilde{v}$ est décrite par le triplet (α, T, K) des paramètres de l'équation (5.16). Afin d'ancrer l'intuition que $-p \geqslant 0$ sans rendre les notation incohérentes avec celles du cas général, nous noterons aussi $\bar{p} = -p \geqslant 0$. Il est important de remarquer que si $p < 0$, le point ζ peut fort bien se trouver à l'intérieur du disque de convergence de v mais hors de celui de $\tilde{v}$.

Notation 6.4. Pour $q \in \mathbb{N}$ et x quelconque, on pose $\Theta_q(x) = \sum_{k=0}^{q-1} x^k$. ♢

Proposition 6.5. Soit $\zeta \in \mathbb{C}$ de module $|\zeta| < \delta(a^{[r]})$. Supposons

$$n > \frac{K}{(1-\alpha|\zeta|)^{T+1}}, \qquad \kappa = 0 \tag{6.4}$$

$$\text{ou} \qquad n > (\alpha\,|\zeta|)^{q/\bar{p}} \left(1 - \left(\frac{K}{(\alpha\,|\zeta|)^{q/\bar{p}} + K + 1}\right)^{\frac{1}{T+1}}\right)^{q/p}, \qquad \kappa < 0. \tag{6.5}$$

Alors pour tout j, le reste d'ordre n de la dérivée j-ième de u satisfait l'inégalité

$$\left|u^{(j)}_{n;}(\zeta)\right| \leqslant \frac{\tilde{v}^{(j)}(r_n)}{q^{(\bar{p}/q)n}\,\Gamma(\frac{n}{q}+1)^{\bar{p}}} \left(\frac{|\zeta|}{r_n}\right)^n h\!\left(\frac{|\zeta|}{r_n}\right) \tag{6.6}$$

où

$$r_n = \frac{1}{\alpha}\left(1 - \left(\frac{K}{n+K-1}\right)^{\frac{1}{T+1}}\right) \qquad h(x) = \frac{\Theta_q(x)}{1-(n+q)^p\,x^q}.$$

(En particulier, si $\kappa = 0$, on a $p = 0$, $q = 1$, et donc $h(x) = 1/(1-x)$.)

Quand n tend vers l'infini, ζ étant fixé, et pour des conditions initiales génériques, la borne (6.6) est optimale à un facteur sous-exponentiel près. ♢

La figure 6.1 illustre le comportement typique de cette borne pour les fonctions entières. Une fois de plus, on peut faire apparaître explicitement le facteur $n!^{p/q}$ dans (6.6) en utilisant le lemme 5.35 du chapitre 5.

DÉMONSTRATION. Revenons un instant sur le sens des expressions qui apparaissent dans l'énoncé de la proposition. On a clairement $r_n < \alpha^{-1}$, donc r_n appartient au disque de convergence de $\tilde{v}^{(j)}$. Pour $\kappa = 0$, la condition (6.4) entraîne de plus $r_n > |\zeta|$; tandis que pour $\kappa < 0$, on a $n > (\alpha\,|\zeta|)^{q/\bar{p}}$ d'après (6.5), d'où

$$(n+q)^p \left(\frac{|\zeta|}{r_n}\right)^q < n^p \left(\frac{|\zeta|}{r_n}\right)^q < \frac{1}{(\alpha\, r_n)^q} < 1. \tag{6.7}$$

Dans les deux cas, l'évaluation de h en $|\zeta|/r_n$ est licite.

Démontrons maintenant l'inégalité (6.6). En utilisant la borne « du col »

$$\tilde{v}_k^{(j)} \leqslant r_n^{-k}\, \tilde{v}^{(j)}(r_n)$$

(attention aux positions respectives des n et des k), il vient

$$\left|u_{n;}^{(j)}(\zeta)\right| \leqslant v_{n;}^{(j)}(|\zeta|) = \sum_{k=n}^{\infty} \frac{\tilde{v}_k^{(j)}}{\psi_k} |\zeta|^k \leqslant \tilde{v}^{(j)}(r_n) \left(\frac{|\zeta|}{r_n}\right)^n \sum_{k=0}^{\infty} \frac{\psi_n}{\psi_{n+k}} \left(\frac{|\zeta|}{r_n}\right)^k.$$

Dans le cas $\kappa = 0$, c'est exactement l'inégalité cherchée. Supposons maintenant $p < 0$. On a vu qu'alors $\psi_n = q^{\bar{p}/q}\,\Gamma(n/q+1)^{\bar{p}}$ croît avec n, de sorte que

$$\begin{aligned} H_n(x) &= \sum_{k=0}^{\infty} \frac{\psi_n}{\psi_{n+k}} x^k \leqslant \sum_{t=0}^{\infty} \sum_{u=0}^{q-1} \frac{\psi_n}{\psi_{n+tq}} x^{tq+u} \\ &= \sum_{u=0}^{q-1} x^u \sum_{t=0}^{\infty} \frac{x^{tq}}{\big((n+q)\,(n+2\,q)\cdots(n+q)\big)^{\bar{p}}}. \end{aligned}$$

Avec $x = |\zeta|/r_n$, on a $n^{\bar{p}} \geqslant x^q$ d'après (6.7), donc $H_n(x) \leqslant h(x)$, d'où (6.6).

La finesse des bornes provient de ce que $\tilde{v}(r_n)\, r_n^{-n} = O(\alpha^{n+o(n)})$ quand $n \to \infty$ (proposition 5.34), tandis que $h(|\zeta|/r_n) \to h(\alpha\,|\zeta|)$. □

EXEMPLE 6.6. Les résultats donnés par mon implémentation sur les fonctions usuelles font souvent intervenir l'une des bornes simplifiées que nous verrons par la suite plutôt que la forme générale. Sur un exemple un peu artificiel, on trouve :

```
> deq := prettify(holexprtodiffeq(erf(1/(2-z^2)), y(z)));
```

$$\left\{(-2+z^2)^3\, z\, \frac{\mathrm{d}^2}{\mathrm{d}z^2} y(z) + (8 - 10\,z^4 + 3\,z^6)\, \frac{\mathrm{d}}{\mathrm{d}z} y(z) = 0,\ y(0) = \operatorname{erf}\left(\frac{1}{2}\right),\ (\mathrm{D}\circ\mathrm{D})(y)(0) = \frac{\mathrm{e}^{\frac{-1}{4}}}{\sqrt{\pi}}\right\}$$

```
> bound_diffeq_tail(deq, y(z), n);
```

$$\begin{cases} \dfrac{228936867129}{250000000000}\, \dfrac{\mathrm{e}^{\left(\frac{1}{2\left(\frac{1}{n+9}\right)^{\frac{2}{3}}}\right)} \left(\frac{1}{2}\, \frac{\sqrt{2}\,|z|}{1-\left(\frac{1}{n+9}\right)^{\frac{1}{3}}}\right)^n}{1 - \frac{1}{2}\, \frac{\sqrt{2}\,|z|}{1-\left(\frac{1}{n+9}\right)^{\frac{1}{3}}}}, & \dfrac{1}{\left(1-\frac{1}{2}\sqrt{2}\,|z|\right)^3} \leq n \\ \dfrac{228936867129}{250000000000}\, \mathrm{e}^{\left(\frac{1}{2\left(1-\frac{1}{2}\sqrt{2}|z|\right)^2}\right)}, & \text{otherwise} \end{cases}$$

La première borne est l'instanciation de (6.6) avec sa condition de validité. Elle est complétée par une borne sur la somme des valeurs absolues des termes de la série sans troncature (voir section suivante), qui couvre les autres cas. ♢

Pour des applications où T est grand et $|\zeta| \approx \alpha^{-1}$, la borne (6.6) ne devient fine que pour n très grand, et il pourrait être intéressant de chercher des majorations plus précises. Des problèmes semblables apparaissent quand K est trop grand.

Remarque 6.7. En §8.5.2, nous ferons ponctuellement appel aux majorants un peu plus généraux $w(z)=v(\eta+z)$ où v est de la forme considérée jusqu'ici et $0<\eta\ll\alpha^{-1}$. On a $v(\eta+z)=w_{;n}(z)+w_{;n}(z)$ et donc, $w_{;n}$ étant un polynôme de degré au plus $n-1$,

$$w_{n;}(z)\preccurlyeq v_{n;}(\eta+z),$$

Si $u(z)\preccurlyeq v(\eta+z)$, il vient ainsi

$$\left|u_{n;}^{(j)}(|\zeta|)\right|\leqslant v_{n;}^{(j)}(\eta+|\zeta|)$$

le membre droit étant donné par la proposition 6.5. ♢

6.2.2 Ordre et type d'une fonction entière

On a parfois besoin de bornes sur les $u_{n;}(z)$ quand les conditions (6.4) et (6.5) ne sont pas satisfaites, et particulièrement de bornes sur $u(\zeta)=u_{0;}(\zeta)$. Pour des fonctions modèles simples, cela correspond à n trop petit pour que $|u_{n;}(\zeta)|$ décroisse (voir figure 6.1). Des bornes indépendantes de n donnent alors des résultats satisfaisants. Demander des bornes fines au sens du théorème 5.1 ou de la proposition 6.5 n'a évidemment plus de sens. Mais la finesse des séries majorantes entraîne tout de même que la borne ne s'éloigne pas trop du pire comportement possible de $u(\zeta)$ quand ζ varie dans le domaine de validité de la borne.

Ce second critère de qualité est particulièrement important dans le cas des fonctions entières, où le domaine de validité est le plan complexe tout entier. On le quantifie à l'aide de l'ordre et du type de le fonction que l'on borne. Les propriétés suivantes sont classiques, voir par exemple [156, chap. 1].

Définition 6.8. Soit f une fonction entière. Pour $r>0$, on note

$$M_f(r)=\sup_{|\zeta|=r}|f(\zeta)|.$$

On appelle *ordre* de f la quantité

$$\rho_f=\limsup_{r\to\infty}\frac{\log\log M_f(r)}{\log r}\in[0,+\infty].$$

Si f est d'ordre fini ρ, on appelle *type* de f la quantité

$$\sigma_f=\limsup_{r\to\infty}\frac{\log M_f(r)}{r^\rho}\in[0,+\infty].$$ ♢

Proposition 6.9. L'ordre et le type d'une fonction f sont reliés à la croissance de ses coefficients de Taylor par

$$\rho_f=\limsup_{n\to\infty}\frac{n\log n}{\log(1/f_n)},\qquad \sigma_f=\frac{1}{\rho\,e}\limsup_{n\to\infty}\left(n\,|f_n|^{\rho_f/n}\right).$$ ♢

Ainsi, l'ordre et le type sont les valeurs optimales ρ,σ telles que

$$|f(\zeta)|\leqslant\exp\left(\sigma\,|\zeta|^\rho+o(|\zeta|^\rho)\right)$$

quand $\zeta\to\infty$. Si $f_n=n!^{-\bar p/q}\alpha^n\,e^{o(n)}$, la proposition précédente donne

$$\rho_f=\frac{q}{\bar p}\qquad\text{et}\qquad\sigma_f=\frac{\bar p}{q}\limsup_{n\to\infty}\left|\alpha\,\frac{n/e}{(n!)^{1/n}}\right|=\frac{\bar p}{q}\,\alpha.$$

Observons qu'une fonction entière D-finie qui n'est pas un polynôme est d'ordre fini et de type normal, c'est-à-dire fini non nul.

6.2.3 Bornes sur les sommes et les restes de petit ordre

LEMME 6.10. Supposons $\kappa<0$. La série $\sum_k \psi_k^{-1} x^k$ est d'ordre $q/\bar{p}$ et de type $\bar{p}/q$, et satisfait

$$\sum_{k=n}^{\infty} \frac{x^k}{\psi_k} \leqslant \Theta_q(x) \exp\left(\frac{\bar{p}}{q}\, x^{q/\bar{p}}\right)$$

pour tous $n\in\mathbb{N}$ et $x\geqslant 0$. ♢

DÉMONSTRATION. Le premier point résulte directement de la proposition 6.9. Comme $\kappa<0$, la suite (ψ_n) est croissante, et l'on a

$$\psi_{qt+u} = \psi_{qt} = q^{\bar{p}t}\, t!^{\bar{p}} \geqslant \left(\frac{q}{\bar{p}}\right)^{\bar{p}t} (\bar{p}\,t)!$$

pour tout $t,u\in\mathbb{N}$. Il s'ensuit que

$$\begin{aligned}\sum_{k=n}^{\infty} \frac{z^{\bar{p}k}}{\psi_k} &\trianglelefteqslant \sum_{k=0}^{\infty} \frac{z^{\bar{p}k}}{\psi_k} = \sum_{u=0}^{q-1}\sum_{t=0}^{\infty} \frac{z^{\bar{p}(qt+u)}}{\psi_{qt+u}} \\ &\trianglelefteqslant \Theta_q(z^{\bar{p}}) \sum_{t=0}^{\infty} \left(\frac{\bar{p}}{q}\right)^{\bar{p}t} \frac{z^{\bar{p}qt}}{(\bar{p}\,t)!} \trianglelefteqslant \Theta_q(z^{\bar{p}}) \exp\left(\frac{\bar{p}}{q}\, z^q\right)\end{aligned}$$

d'où le résultat en évaluant en $z=x^{1/\bar{p}}$. □

PROPOSITION 6.11. On a

$$\left|u_{n;}^{(j)}(\zeta)\right| \leqslant \begin{cases} v^{(j)}(|\zeta|), & \kappa=0 \\ \tilde{v}^{(j)}(r) \exp\left(\dfrac{\bar{p}}{q}\left(\dfrac{|\zeta|}{r}\right)^{q/\bar{p}}\right) \Theta_q\left(\dfrac{|\zeta|}{r}\right), & \kappa<0 \end{cases}$$

pour tous $n\in\mathbb{N}$, $j\in\mathbb{N}$, $r\in\,]0;\,\alpha^{-1}[$ et pour tout ζ appartenant au domaine de convergence de v. ♢

DÉMONSTRATION. Pour $\kappa=0$, le résultat est évident. Supposons $\kappa<0$: on obtient

$$\left|u_{n;}^{(j)}(\zeta)\right| \leqslant \tilde{v}^{(j)}(r) \sum_{k=n}^{\infty} \frac{1}{\psi_k}\left(\frac{|\zeta|}{r}\right)^k \leqslant \tilde{v}^{(j)}(r) \exp\left(\frac{\bar{p}}{q}\left(\frac{|\zeta|}{r}\right)^{q/\bar{p}}\right) \sum_{u=0}^{q-1}\left(\frac{|\zeta|}{r}\right)^u$$

en combinant le lemme 6.10 avec la majoration habituelle $v_k^{(j)} \leqslant \psi_k^{-1} r^{-k} \tilde{v}^{(j)}(r)$. □

Dans le cas d'une fonction entière, la borne de la proposition 6.11 respecte l'ordre de la fonction, et peut être rendue aussi proche que l'optimum vis-à-vis de son type, sans l'atteindre. Au moins dans le cas simple $T=0$, on peut se débarrasser de cette restriction. (Ce résultat répond à une question que m'a posée Sylvain Chevillard. Il cherchait à utiliser des bornes comme celle de l'exemple 6.12 ci-dessous afin de contrôler l'accumulation d'erreur dans l'évaluation en virgule flottante de fonctions entières, en l'occurrence les fonctions d'Airy Ai et Bi.)

EXEMPLE 6.12. La fonction d'Airy Bi se comporte à l'infini comme [7, formule 10.4.63]

$$\mathrm{Bi}\, x \underset{x\to\infty}{\sim} \frac{\exp\left(\frac{2}{3}\, x^{3/2}\right)}{\sqrt{\pi\, x}}.$$

En utilisant NumGfun, on calcule :

```
> deq := holexprtodiffeq(AiryBi(z), y(z));
```

$$\left\{\frac{\mathrm{d}}{\mathrm{d}z}\left(\frac{\mathrm{d}}{\mathrm{d}z}y\left(z\right)\right)-y\left(z\right)z, y\left(0\right)=\frac{1}{3}\frac{3^{\frac{5}{6}}}{\Gamma\left(\frac{2}{3}\right)}, \left(y'\right)\left(0\right)=\frac{1}{2}\frac{3^{\frac{2}{3}}\Gamma\left(\frac{2}{3}\right)}{\pi}\right\}$$

```
> bound_diffeq_tail(deq, y(z), 0);
```

$$\frac{11440000000000}{53514763737}\frac{|z|^{12}-1}{|z|-1}+\frac{5000000}{3720087}\frac{|z|^{12}\left(|z|^{3}-1\right)\mathrm{e}^{\left(\frac{2}{3}|z|^{\frac{3}{2}}\right)}}{|z|-1}$$

soit une borne optimale à un facteur polynomial près. ♢

Proposition 6.13. Supposons $\kappa<0$ et $T=0$. Soit $s\in\mathbb{N}$ tel que $s\,\bar{p}\geqslant K-1$. On a alors

$$|v(\zeta)|\leqslant|v_{;qs}(\zeta)|+M\,|\alpha\,\zeta|^{qs}\,\Theta_q(\alpha\,|\zeta|)\exp\left(\frac{\bar{p}}{q}|\alpha\,\zeta|^{q/\bar{p}}\right),$$

où

$$M\leqslant\frac{1}{s!^{\bar{p}}}\frac{(s\,(\bar{p}+q))!}{(K-1)!\,(s\,q)!}.$$ ♢

Démonstration. On a dans ce cas

$$v(z)=\sum_{n=0}^{\infty}\binom{n+K-1}{K-1}\frac{(\alpha\,z)^n}{\psi_n}.$$

Si M est tel que pour $n\geqslant s\,q$,

$$\binom{n+K-1}{K-1}\frac{\psi_{n-sq}}{\psi_n}=\frac{1}{(K-1)!}\frac{(n+1)^{\uparrow(K-1)}}{\left(n\left(n-q\right)\cdots\left(n-(s-1)\,q\right)\right)^{\bar{p}}}\leqslant M,$$

alors $v_{qs;}(z)\leqslant M\sum_{n=0}^{\infty}\psi_n^{-1}\,(\alpha\,z)^{n+qs}$, d'où

$$v_{qs;}(|\zeta|)\leqslant M\,|\alpha\,\zeta|^{qs}\,\Theta_q(|\alpha\,\zeta|)\exp\left(\frac{\bar{p}}{q}\,|\alpha\,\zeta|^{q/\bar{p}}\right)$$

d'après le lemme 6.10. Les inégalités $s\,\bar{p}\geqslant K-1$ et $n\geqslant s\,q$ entraînent

$$\begin{aligned}\frac{(n+1)^{\uparrow(K-1)}}{\left((n/q)^{\downarrow s}\right)^{\bar{p}}}&=\left(\frac{q}{n}\right)^{s\bar{p}}n^{K-1}\frac{\left(1+\frac{1}{n}\right)\left(1+\frac{2}{n}\right)\ldots\left(1+\frac{K-1}{n}\right)}{\left(\left(1-\frac{q}{n}\right)\left(1-\frac{2\,q}{n}\right)\cdots\left(1-\frac{(s-1)\,q}{n}\right)\right)^{\bar{p}}}\\&\leqslant q^{s\bar{p}}\frac{\left(1+\frac{1}{s\,q}\right)\left(1+\frac{2}{s\,q}\right)\ldots\left(1+\frac{s\,\bar{p}}{s\,q}\right)}{\left(1\left(1-\frac{1}{s}\right)\left(1-\frac{2}{s}\right)\cdots\left(1-\frac{(s-1)}{s}\right)\right)^{\bar{p}}}\\&=\frac{(s\,q+1)\,(s\,q+2)\cdots(s\,q+s\,\bar{p}\,)}{(s\,(s-1)\,(s-2)\cdots1)^{\bar{p}}}=\frac{(s\,(\bar{p}+q))!}{(s\,q)!\,s!^{\bar{p}}}\end{aligned}$$

ce qui montre que l'on peut prendre M comme indiqué. □

6.2.4 Forme close pour une singularité dominante régulière

Dans le cas important $\kappa=T=0$, c'est-à-dire quand l'équation différentielle (6.3) possède une singularité dominante finie régulière, les restes $v_{n;}(z)$ des séries majorantes que nous avons adoptées admettent des expressions closes relativement simples.

Proposition 6.14. Supposons $\kappa=T=0$ et $K\in\mathbb{N}$ dans la sortie de l'algorithme 5.30. On a alors pour tout n

$$v_{n;}(z)=(\alpha\,z)^n\,p(n)\tag{6.8}$$

où

$$p(n) = \binom{n+K-1}{K-1}\, {}_2\mathrm{F}_1\left(\begin{matrix}1 & n+K\\ & n+1\end{matrix}\middle|\, \alpha z\right) \in \mathbb{Q}(\alpha\, z)[n]$$

est un polynôme en n de degré K explicite. ♢

DÉMONSTRATION. On part de l'égalité

$$v_{n;}(z) = ((1-\alpha\, z)^{-K})_{n;} = (\alpha\, z)^n \binom{n+K-1}{K-1} \sum_{k=0}^{\infty} \frac{(n+k+1)^{\uparrow(K-1)}}{(n+1)^{\uparrow(K-1)}} (\alpha\, z)^k.$$

L'expression hypergéométrique de p s'en déduit immédiatement. Pour obtenir la forme polynomiale, on pose à K fixé

$$(n+k+1)^{\uparrow(K-1)} = \sum_{i=1}^{K} c^{[i]}(n) \binom{k-1}{i-1},$$

et alors

$$p(n) = \frac{1}{(K-1)!} \sum_{k=0}^{\infty} \sum_{i=1}^{K} c^{[i]}(n) \binom{k-1}{i-1} (\alpha\, z)^k = \frac{1}{(K-1)!} \sum_{i=1}^{K} \frac{c^{[i]}(n)}{(1-\alpha\, z)^i}$$

est le développement explicite de p. □

La forme polynomiale permet d'afficher des bornes comme celle de l'exemple 6.1 ou encore la suivante, en général plus lisibles que la forme générale (6.6). Cependant, l'évaluation de ces formules peut présenter des problèmes d'instabilité numérique en raison d'annulations entre les termes. Pour coder en dur la formule (6.8) dans un système capable d'évaluer les fonctions hypergéométriques, l'expression en termes de ${}_2\mathrm{F}_1$ est préférable.

EXEMPLE 6.15. Pour l'équation différentielle d'ordre deux tirée au hasard

```
> deq := {(7/15-13/30*z+9/10*z^2)*y(z)+(2/15-11/60*z+1/60*z^2)*diff(y(z),z)+(9/10-
  13/20*z+13/30*z^2)*diff(diff(y(z),z),z), y(0) = -3/10, D(y)(0) = -11/12};
```

$$\left\{\left(\frac{7}{15}-\frac{13}{30}z+\frac{9}{10}z^2\right) y(z) + \left(\frac{2}{15}-\frac{11}{60}z+\frac{1}{60}z^2\right)\left(\frac{\mathrm{d}}{\mathrm{d}z}y(z)\right) + \left(\frac{9}{10}-\frac{13}{20}z+\frac{13}{30}z^2\right)\left(\frac{\mathrm{d}}{\mathrm{d}z}\left(\frac{\mathrm{d}}{\mathrm{d}z}y(z)\right)\right), y(0) = \frac{-3}{10}, (y')(0) = \frac{-11}{12}\right\}$$

on obtient la borne

```
> bound_diffeq_tail(deq, y(z), n);
```

$$\frac{470297685171}{100000000000}\, \frac{\sqrt{39}\left(\frac{1}{9}|z|\sqrt{39}\right)^n}{3\sqrt{39}-13|z|}$$

valable pour tout n. ♢

6.3 Application à l'approximation de fonctions D-finies

6.3.1 Nombre de termes à sommer

Au sein d'algorithmes d'évaluation numérique comme celui du chapitre 8, les bornes sur les restes servent avant tout à décider à quel ordre n tronquer le développement en série d'une fonction y pour calculer $y(z)$ à une précision ε donnée près. (Les bornes sur $y(z) = y_{0;}(z)$ ont aussi leur utilité, comme nous le verrons en §8.3.) La proposition 6.5 se traduit comme suit.

Rayon de convergence et singularité dominante	Borne sur le reste ($\beta<1$)	Nombre de termes pour précision 2^{-d}
$\rho<\infty$, cas général	$A\left(\frac{\lvert z\rvert}{\rho}\right)^n \exp\left(K\,n^\beta\right)$	$\frac{d}{\log_2 \frac{\rho}{\lvert z\rvert}}+o(d)$
$\rho<\infty$, point singulier régulier	$A\left(\frac{\lvert z\rvert}{\rho}\right)^n n^K$	$\frac{d}{\log_2 \frac{\rho}{\lvert z\rvert}}+o(d)$
$\rho=\infty$	$A\,\frac{(\lvert z\rvert/\rho)^n}{n!^{\bar p/q}}\exp\left(K\,n^\beta\right)$	$\frac{q}{\bar p}\,\frac{d}{\log_2 d}+o\!\left(\frac{d}{\log d}\right)$

Tableau 6.1. Bornes sur les restes de séries et ordres de troncature correspondants pour garantir une précision donnée.

Corollaire 6.16. Soit u une fonction D-finie solution d'une équation dont le comportement générique des solutions — au sens de la section 3 — est donné par les paramètres $\kappa=p/q=-\bar p\,/q$ et $\alpha=\rho^{-1}$, avec par convention $\bar p=0$ et $q=1$ si $\kappa=0$. On peut calculer une borne $b(\zeta,n)$ telle que

$$\forall \zeta\neq 0,\quad \forall n,\quad \log\lvert u_{n;}(\zeta)\rvert\leqslant b(\zeta,n), \tag{6.9}$$

et, pour $\lambda<1$ fixé,

$$b(\zeta,n)=-n\left(\log\frac{\rho}{\lvert\zeta\rvert}+\frac{\bar p}{q}\,(1+\log n)\right)+o(n)$$

uniformément en $\zeta\in\bar{\mathcal{D}}(0,\lambda\,\rho)\setminus\{0\}$. ♢

Démonstration. La borne (6.6) entraîne

$$\log\lvert u_{n;}(\zeta)\rvert\leqslant\log\left(\tilde v_j(r_n)\,r_n^{-n}\right)-\log\left(q^{(\bar p/q)n}\,\Gamma(\tfrac{n}{q}+1)^{\bar p}\right)+n\log\lvert\zeta\rvert+\log\big(h(\lvert\zeta\rvert/r_n)\big).$$

On a $\log\left(\tilde v_j(r_n)\,r_n^{-n}\right)=\rho^{-1}\,n+o(n)$ d'après la proposition 5.34, et

$$\log\left(q^{(\bar p/q)n}\,\Gamma\left(\frac{n}{q}+1\right)^{\bar p}\right)=\frac{\bar p}{q}\,n\,(\log n+1+o(1))$$

d'après la formule de Stirling. Le terme en $h(\lvert\zeta\rvert/r_n)$ est borné pour $\lvert\zeta\rvert\leqslant\lambda\,\rho$. □

Le tableau 6.1 récapitule, suivant les caractéristiques de l'équation différentielle, les ordres de grandeur des bornes (6.9) et les nombres de termes à retenir correspondants. L'estimation automatique précise du nombre de termes à sommer ouvre la voie à une analyse fine, « sans $O(\cdot)$ » de la complexité des algorithmes d'évaluation. En outre, sauf solution non-générique, les ordres de troncature calculés à partir du corollaire 6.16 sont optimaux à un terme sous-linéaire en n près, ce qui se reflète dans la complexité.

6.3.2 Assouplissement : calcul semi-numérique des majorants

Les algorithmes de calcul de bornes que nous avons développés sont donc utilisables au sein d'un programme d'évaluation numérique des fonctions D-finies. Cependant, ce contexte diffère de celui des bornes symboliques calculées pour elles-mêmes, sur plusieurs plans :

(i) l'objectif n'est plus d'obtenir des bornes *lisibles* ou *réutilisables* mais des bornes *aisées à évaluer* en un point z et un ordre n donnés ;

(ii) la finesse n'a guère d'intérêt que si son surcoût en temps de calcul est nettement plus faible que le temps qu'elle fait gagner dans l'évaluation proprement dite ;

(iii) dans l'algorithme de prolongement analytique du chapitre 8, il est commode de majorer uniformément toute une base de solutions d'une équation donnée ;

(iv) et pour ce qui est de l'extension aux points singuliers réguliers du même algorithme, la base en question est formée non plus de fonctions analytiques mais de séries logarithmiques.

Le point (iii) ne présente pas de difficulté. On a simplement l'opportunité de calculer les paramètres de la borne une fois pour toutes, en ajustant les conditions initiales de l'équation majorante pour majorer simultanément celles de tous les éléments de la base considérée. Nous nous occuperons du point (iv) en §6.4.

Les points (i) et (ii) sont liés, et nous pouvons tirer parti de la liberté laissée par le premier pour améliorer le second. Les manipulations de nombres algébriques dans les algorithmes du chapitre précédent, si elles semblent inévitables pour satisfaire le critère de finesse, sont complexes et coûteuses. Tant pour limiter le temps de calcul des bornes que pour faciliter leur implémentation à plus bas niveau, il est souhaitable d'avoir une variante approchée qui ne sacrifie pas la finesse pratique.

On remplace α par une approximation rationnelle $\tilde{\alpha} \geqslant \alpha$ dans l'algorithme 5.26 (bornes sur les équations différentielles normalisées) et les appels à l'algorithme 5.22 (bornes sur les fractions rationnelles) qu'il effectue. En toute rigueur, la sortie n'est donc plus une borne fine au sens de la définition 5.20. Le point crucial, qui fait la différence avec les majorants habituels n'atteignant pas le rayon de convergence de la série majorée, est que l'on conserve cependant la forme générale des majorants, sans « cacher » de facteur sous-exponentiel dans le rapport $(\alpha/\tilde{\alpha})^n$.

Plus précisément, si l'on dispose d'un solveur polynomial boîte noire[4], la ligne 4 de l'algorithme 5.26 devient

4.1 calculer des approximations par défaut de $\delta(P_\alpha)$ et des modules des pôles des $\tilde{a}^{[k]}$

4.2 en déduire un rationnel $\tilde{\mu} \leqslant \min\Big(\delta(P_\alpha), \delta\big(\operatorname{den}\tilde{a}^{[0]}\big), \ldots, \delta\big(\operatorname{den}\tilde{a}^{[r-1]}\big)\Big)$

4.3 $T := \max\big(s\{0\} \cup \{\nu_k - r + k : 0 \leqslant k < r\}\big)$, où $\nu_k \geqslant \nu_\delta\big(\operatorname{den}\tilde{a}^{[k]}\big)$ est le nombre total, avec multiplicités, de racines de $\operatorname{den}\tilde{a}^{[k]}$ « proches » de $\tilde{\mu}$ d'après un test numérique

tandis que $\delta(P_\alpha)$ est remplacé par $\tilde{\mu}$ (et P_α par $X - \tilde{\mu}$) dans toute la suite. Dans l'algorithme 5.22, les expressions $\delta(D_i)$ qui apparaissent aux étapes 7 et 8 sont remplacées par *les approximations calculées à la ligne 4.1 ci-dessus*, de façon à avoir $\delta(P_\alpha) \geqslant \delta(D_i)$ pour tout i. En particulier, l'étape 8, tout comme la nouvelle étape 4 de l'algorithme 5.26, ne requiert plus de test d'égalité exact entre algébriques.

EXEMPLE 6.17. Comparons les bornes classiques et leur variante « numérique » pour une fonction très simple :

```
> deq := holexprtodiffeq(arctan(z)^2, y(z));
```

$$\left\{(2z+2z^3)\left(\frac{\mathrm{d}}{\mathrm{d}z}y(z)\right) - 2 + (1+2z^2+z^4)\left(\frac{\mathrm{d}}{\mathrm{d}z}\left(\frac{\mathrm{d}}{\mathrm{d}z}y(z)\right)\right), y(0)=0, (y')(0)=0\right\}$$

```
> bound_diffeq(deq, y(z));
```

$$\frac{166666666867}{1000000000000}\,\frac{1}{(1-z)^3}$$

```
> kernelopts(opaquemodules=false): NumGfun:-numeric_mode := true:  # non documenté
> bound_diffeq(deq, y(z));
```

$$\frac{166666660367}{1000000000000}\,\frac{1}{\left(1-\frac{50000000}{49999999}z\right)^3}$$

Le rayon de convergence du second majorant est strictement plus petit que celui des fonctions majorées ou du premier majorant, mais la puissance 3 subsiste et le facteur

4. On peut sinon se contenter de minorer les modules des racines de chaque facteur des décompositions sans carré qui interviennent.

constant n'a pas augmenté. ♢

6.3.3 Expériences

Le tableau 6.2 présente les résultats d'expériences relatives à la finesse des bornes calculées par NumGfun pour les ordres de troncature de développements de Taylor de quelques fonctions simples.

Une étiquette de colonne de la forme $f(a)$ ou $f(z)\,@\,a$ fait référence à une équation différentielle (accompagnée de conditions initiales en zéro) satisfaite par f et un point d'évaluation de module strictement inférieur à celui des singularités dominantes de l'équation. Dans chaque cas, on a calculé un nombre de termes du développement en série entière de f à l'origine qui suffit à déterminer $f(a)$ à 10^{-10}, 10^{-100} et 10^{-1000} près. Ce nombre de termes est obtenu en combinant l'algorithme de calcul de séries majorantes et les formules sur les restes de la section 6.2, puis en recherchant par dichotomie le plus petit ordre de troncature qui garantit la précision cherchée. Les cellules internes du tableau comparent le nombre de termes calculé au minimum convenable, obtenu quant à lui par recherche exhaustive.

Par exemple, la colonne « $\mathrm{erf}^2(1)$ » correspond à l'évaluation en $z=1$ de la fonction $u(z)=\mathrm{erf}(z)^2$ représentée comme l'unique solution de

$$(2+8\,z^2)\,u'(z)+6\,z\,u''(z)+u'''(z)=0, \qquad u(0)=0, \quad u'(0)=0, \quad u''(0)=\frac{8}{\pi}.$$

L'algorithme de calcul de séries majorantes a permis de déterminer que

$$|u_{;190}(1)-u(1)|\leqslant 10^{-100}$$

mais il se trouve que seuls les 163 premiers de ces 190 termes sont réellement nécessaires. On observe que les bornes ne s'éloignent pas trop des valeurs optimales, et d'autant moins que la précision demandée est grande.

Les fonctions testées sont divisées en trois classes, qui correspondent aux principaux types de comportements asymptotiques que peut présenter la suite des coefficients d'une série D-finie convergente. Ces comportements sont eux-mêmes caractérisés (pour des conditions initiales génériques) par la nature des singularités dominantes de l'équation différentielle :

1. points singuliers réguliers uniquement ($\kappa=0$ et $T=0$ avec les notations de l'algorithme 5.30) ;
2. points singuliers irréguliers à distance finie ($\kappa=0$, $T>0$) ;
3. point singulier irrégulier à l'infini ($\kappa<0$).

Une équation dont l'unique point singulier serait un point singulier régulier à l'infini n'aurait que des solutions polynomiales ; et la situation $\kappa>0$ est exclue car elle correspond à une série divergente. Les fonctions de la deuxième classe comportent toutes des précompositions par des fractions rationnelles car les équations du monde des fonctions spéciales n'ont le plus souvent pas plus de deux points singuliers irréguliers, or l'habitude est de placer ceux-ci à l'infini et à l'origine.

Dans chaque classe, les trois derniers exemples illustrent ce qu'il se passe quand $|a|$ approche le rayon de convergence de la série. Observons cependant que sommer un développement en série à un ordre très élevé n'est pas une façon recommandable d'évaluer une fonction D-finie en un tel a. On peut éviter l'explosion du nombre de termes à sommer (optimal comme calculé) en remplaçant l'évaluation directe de f en a par plusieurs pas de prolongement analytique, suivant une méthode proposée par Chudnovsky et Chudnovsky [54, §4] et exposée en §8.3 de ce mémoire (voir notamment l'exemple 8.7).

Enfin, l'exemple du sinus intégral $\mathrm{Si}(z)$ présente une caractéristique intéressante : l'origine est un point singulier régulier de l'équation différentielle associée, et cependant $\mathrm{Si}(z)$ y est analytique et peut être définie par des conditions initiales simples. Ainsi, l'algorithme s'applique directement, même sans les extensions à venir pour traiter les solutions séries généralisées.

Singularité dominante régulière					
	$\frac{1}{(1-z)^2}$ @ $\frac{1}{2}$	$\frac{\cos z}{1-z}$ @ $\frac{1}{2}$	$\frac{\cos z}{1-z^2}$ @ $\frac{1}{2}$	$\frac{\cos z}{(1-z)^2}$ @ $\frac{1}{2}$	$\frac{(z+1)^2\cos z}{(z^3+z+1)^2}$ @ $\frac{1}{10}$
10^{-10}	40/40	46/34	54/33	54/39	24/12
10^{-100}	342/342	350/333	364/331	364/341	140/121
10^{-1000}	3336/3335	3346/3323	3366/3321	3366/3334	1232/1201
	$\frac{\operatorname{arccot}(z)}{(z^2-1)(z^2+5)}$ @ $\frac{1}{2}$	$\psi(1/2)$	$\arctan\frac{1}{2}$	$\arctan\frac{9}{10}$	$\arctan\frac{99}{100}$
10^{-10}	64/27	40/23	44/28	336/164	4238/1496
10^{-100}	380/321	342/313	348/324	2338/2108	25210/21848
10^{-1000}	3392/3307	3336/3293	3344/3310	22050/21754	231844/227810

Singularité dominante irrégulière finie					
	$\cos\frac{z}{1-z}$ @ $\frac{1}{3}$	$\sin\frac{z}{1-z}$ @ $\frac{1}{3}$	$\exp\frac{z}{(1-z)^2}$ @ $\frac{1}{2}$	$\exp\frac{z}{1-z^2}$ @ $\frac{1}{2}$	$\operatorname{erf}\left(\frac{1+z}{2z^2-1}\right)$ @ $\frac{1}{9}$
10^{-10}	48/25	46/24	118/79	68/42	28/12
10^{-100}	290/224	290/225	558/497	416/364	244/132
10^{-1000}	2416/2150	2416/2149	4154/4001	3566/3432	2384/1292
	$\frac{\exp(1/(1-z))}{(1-z)}$ @ $\frac{1}{2}$	$\operatorname{Bi}\left(\frac{1}{1-z}\right)$ @ $\frac{1}{2}$	$\operatorname{Ai}\left(\frac{1}{1-z}\right)$ @ $\frac{1}{2}$	$\operatorname{Ai}\left(\frac{1}{1-z}\right)$ @ $\frac{3}{4}$	$\operatorname{Ai}\left(\frac{1}{1-z}\right)$ @ $\frac{7}{8}$
10^{-10}	70/54	148/56	142/30	1558/77	23818/215
10^{-100}	418/387	664/416	660/345	3430/879	29258/2025
10^{-1000}	3568/3490	4700/3645	4694/3406	16284/8372	69594/18529

Singularité dominante irrégulière infinie					
	$\operatorname{Ai}(4i+4)$	$\operatorname{Bi}(4i+4)$	$\operatorname{Si}(1)$	$\cos(1)$	$\sin(1)$
10^{-10}	92/59	92/59	16/12	18/13	18/14
10^{-100}	226/200	226/200	74/68	76/69	74/70
10^{-1000}	1054/1031	1054/1031	454/448	456/449	456/450
	e^{-100}	$\operatorname{erf}^2(1)$	$\operatorname{erf}(1)$	$\operatorname{erf}(10)$	$\operatorname{erf}(100)$
10^{-10}	298/291	60/33	36/24	628/574	54492/54388
10^{-100}	456/450	190/163	150/138	936/894	54904/54800
10^{-1000}	1406/1402	1036/1011	908/898	2828/2800	58870/58772

Tableau 6.2. Nombre de termes calculé et nombre de termes minimal requis afin d'approcher une fonction en un point et à une précision absolue donnés en tronquant sa série de Taylor à l'origine. Dans ce tableau, Ai et Bi sont les fonctions d'Airy, erf la fonction d'erreur, Si la fonction sinus intégral, et ψ représente la solution de l'équation des ondes sphéroïdale

$$(1-z^2)\,\psi''(z) - 2\,(b-1)\,\psi'(z) + (c - 4\,q\,z^2)\,\psi(z) = 0$$

correspondant au choix de paramètres $b = 1/2$, $q = 1/3$, $c = 1$ et aux conditions initiales $\psi(0) = 1$, $\psi'(0) = 0$. Ces résultats sont repris de Mezzarobba et Salvy [175], et peuvent donc ne pas correspondre exactement à ceux donnés par la dernière version de NumGfun.

6.4 Extension aux séries logarithmiques

Les algorithmes de majoration vus jusqu'ici s'appliquent aux solutions *séries* au voisinage de points singuliers réguliers. J'esquisse pour conclure ce chapitre une extension qui couvre les solutions généralisées

$$u(z) = z^\lambda \sum_{k=0}^{t-1} \varphi_k(z)\,\frac{\log^k z}{k!}. \tag{6.10}$$

Un exemple d'application est l'approximation garantie de fonctions comme les fonctions de Bessel à partir de leurs développements convergents au voisinage de singularités. Contrairement aux précédentes, cette méthode n'a pas encore été implémentée. Des ajustements s'avéreront sans doute nécessaires pour aboutir à de bons résultats en pratique.

Comme en §5.3.3, considérons une équation différentielle avec un point régulier à l'origine, écrite sous la forme

$$L(z,\theta)\cdot u = \Big(Q(\theta) - z\,(\tilde a^{[r-1]}\,\theta^{r-1} + \cdots + \tilde a^{[1]}\,\theta + \tilde a^{[0]})\Big)\cdot u = 0 \tag{6.11}$$

et supposons disposer de majorations des coefficients de la forme

$$\forall i, \quad \tilde{a}^{[i]} \preccurlyeq \binom{r-1}{i} \frac{M}{(1-\alpha z)^{r-i+T}}.$$

Soit

$$u(z) = \sum_{n \in \lambda+\mathbb{N}} \sum_{k=0}^{t} u_{n,k} z^n \frac{\log^k z}{k!} \qquad (\lambda \in \mathbb{C})$$

une solution de (6.11). Rappelons que toute solution n'est pas de cette forme, mais que celles de la base canonique définie en §4.4.3 (p. 72) le sont.

D'après la proposition 4.16 (p. 70)[5], la suite double $\boldsymbol{u} = (u_{n,k})_{n \in \lambda+\mathbb{Z}, k \in \mathbb{N}}$ satisfait la récurrence généralisée *« non bornée »*

$$S_k^{\mu(n)} \cdot \boldsymbol{u} = -T_n(S_k)\, S_n^{-1} \sum_{i=0}^{r-1} \tilde{a}^{[i]}(S_n^{-1})\, (n+S_k)^i \cdot \boldsymbol{u} \tag{6.12}$$

où, $\mu(n)$ désignant la multiplicité de n en tant que racine de Q,

$$T_n(S_k) = \sum_{i \geqslant 0}^{\text{(finie)}} t_i(n)\, S_k^i = \sum_{i \geqslant 0}^{\text{(finie)}} \frac{1}{i!} \frac{\partial^i}{\partial X^i} \frac{X^{\mu(n)}}{Q(X)} \bigg|_{X=n} S_k^i.$$

On pose aussi comme d'habitude $E = \{(n,k) \in \mathbb{C} \times \mathbb{N} : 0 \leqslant k < \mu(n)\}$. L'idée est maintenant de trouver une suite $(v_\nu)_{\nu \in \mathbb{N}}$ pour laquelle on puisse établir par récurrence, à partir de la relation (6.12), la majoration $|u_{\lambda+\nu,k}| \leqslant v_\nu$.

Lemme 6.18. Pour $\lambda \in \mathbb{C}$ fixé, la suite $(\beta(\nu))_{\nu \in \mathbb{N}}$ définie par

$$\beta(\nu) = \nu\, (\nu + |\lambda| + 1)^{r-1} \sum_i |t_i(\nu+\lambda)|$$

est bornée. ♢

Démonstration. Supposons d'abord $Q(\lambda+\nu) \neq 0$, c'est-à-dire $\mu(\lambda+\nu) = 0$. On vérifie par récurrence sur i que $t_i(n)$ coïncide pour ces valeurs de n avec une fraction rationnelle en n de degré au plus $(-r-v)$. Restent les indices ν tels que $\mu(\lambda+\nu) > 0$, pour lesquels le degré peut dépasser $-r$, mais ils sont en nombre fini. □

Soit $C \geqslant 0$ telle que $|\beta(\nu)| \leqslant C$ pour tout $\nu \in \mathbb{N}$. (Sur tout intervalle $[\lambda+i, \lambda+j]$ qui ne contient pas de racine de Q et pour tout v, la fonction

$$\nu \mapsto \left| \nu\, (\nu + |\lambda| + 1)^{r-1}\, t_v(\nu+\lambda) \right|^2$$

est une fonction rationnelle dérivable, dont le maximum se trouve donc à l'une des extrémités de l'intervalle ou en un zéro de sa dérivée. Cela fournit une manière de calculer un tel C.) Le lemme suivant prend la place du lemme 5.24 page 92.

Lemme 6.19. Soit $b \in \mathbb{Q}[[z]]$ une série telle que

$$\forall \nu \in \mathbb{N}, \quad \forall j \in [\![0, \nu-1]\!], \quad \sum_{i=0}^{r-1} |\tilde{a}^{[i]}_{\nu-1-j}|\, (j + |\lambda| + 1)^k \leqslant (\nu + |\lambda| + 1)^{r-1}\, b_{\nu-1-j}.$$

Soit v une solution de l'équation différentielle $v'(z) = C\, b(z)\, v(z)$ telle que $|u_{\lambda+\nu}, k| \leqslant v_\nu$ pour $(\lambda+\nu, k) \in E$. On a alors

$$\forall k \in \mathbb{N}, \quad \sum_{\nu \in \mathbb{N}} u_{\lambda+\nu,k}\, z^\nu \preccurlyeq v(z).$$

5. Appliquée avec $R(z,\theta)\, z = z\, (\tilde{a}^{[r-1]}\, \theta^{r-1} + \cdots + \tilde{a}^{[0]})$, cette convention étant plus commode ici.

Autrement dit, les séries entières φ_k qui apparaissent dans l'écriture (6.10) de $u(z)$ sont uniformément majorés par $v(z)$. ♢

DÉMONSTRATION. Commençons par développer

$$\begin{aligned} S_n^{-1}\sum_{i=0}^{r-1}\tilde{a}^{[i]}(S_n^{-1})\,(n+S_k)^i &= S_n^{-1}\sum_{p=0}^{r-1}\sum_{j=0}^{\infty}\tilde{a}_j^{[p]}\,S_n^{-j}\,(n+S_k)^p \\ &= \sum_{j=0}^{\infty}\sum_{p=0}^{r-1}\tilde{a}_j^{[p]}\,(n-j-1+S_k)^p\,S_n^{-j-1} \\ &= \sum_{j=0}^{\infty}\sum_{p=0}^{r-1}\sum_{q=0}^{p}\binom{p}{q}\tilde{a}_j^{[p]}\,(n-j-1)^q S_k^{p-q}\,S_n^{-j-1}. \end{aligned}$$

En injectant cette égalité dans (6.12), on obtient

$$u_{n,k+\mu(n)} = \sum_{v\geqslant 0}^{\text{(finie)}} t_v(n)\sum_{j=0}^{\infty}\sum_{p=0}^{r-1}\sum_{q=0}^{p}\binom{p}{q}\tilde{a}_j^{[p]}\,(n-j-1)^q\,u_{\substack{n-1-j,\\ k+p-q+v}}.$$

Nous allons maintenant montrer par récurrence sur ν que

$$\forall \nu\in\mathbb{Z},\quad \forall k\in\mathbb{N},\quad |u_{\lambda+\nu,k}|\leqslant v_\nu.$$

C'est vrai pour ν suffisamment négatif au vu de la forme générale des solutions, ainsi que pour $(\nu+\lambda,k)\in E$ par hypothèse. Soit $\nu\in\mathbb{Z}$, et supposons $|u_{\lambda+\nu',k}|\leqslant v_{\nu'}$ pour $\nu'<\nu$. En particulier, on a

$$\forall j\in\mathbb{N},\quad \forall k\in\mathbb{N},\quad |u_{\substack{\lambda+\nu-j-1,\\ k+p-q+v}}|\leqslant v_{\nu-j-1}$$

d'où, pour $k\geqslant\mu(\nu+\lambda)$,

$$\begin{aligned} |u_{\nu+\lambda,k}| &\leqslant \sum_{i\geqslant 0}|t_i(n)|\sum_{j=0}^{\infty}\sum_{p=0}^{r-1}\sum_{q=0}^{p}\binom{p}{q}|\tilde{a}_j^{[p]}|\,|\nu+\lambda-j-1|^q\,v_{\nu-1-j} \\ &= \sum_{i\geqslant 0}|t_i(n)|\sum_{j=0}^{n-1}\sum_{p=0}^{r-1}|\tilde{a}_j^{[p]}|\,(|\nu+\lambda-j-1|+1)^p\,v_{\nu-1-j} \\ &\leqslant \sum_{i\geqslant 0}|t_i(n)|\sum_{j=0}^{\nu-1}(\nu+|\lambda|+1)^{r-1}\,b_j\,v_{\nu-1-j} \\ &\leqslant \frac{C}{n}\sum_{j=0}^{n-1}b_j\,v_{\nu-1-j} = v_\nu. \end{aligned}$$

Au final, on a $|u_{\nu+\lambda,k}|\leqslant v_\nu$ pour tous ν et k. □

On aboutit au pendant suivant de la proposition 5.27.

PROPOSITION 6.20. Supposons donnés un opérateur différentiel D, régulier à l'origine, et la décomposition sur la base canonique à l'origine (définition 4.20 p. 72) d'une solution u de $D\cdot u=0$. Écrivons $u=\sum_i z^{\lambda_i}\,\varphi_i(z)\log^{k_i}z$, avec $\varphi_i\in\mathbb{C}[[z]]$. On peut calculer des paramètres de bornes $T\in\mathbb{N}$, $\alpha\in\mathbb{R}_+$, $K_i\in\mathbb{N}^*$ et $A_i\in\mathbb{Q}_+$ tels que

$$\forall i,\quad \varphi_i(z)\preccurlyeq\begin{cases} A_i\,(1-\alpha\,z)^{-K_i} & \text{si } T=0\\ A_i\exp\Big(K_i\,T^{-1}\,(1-\alpha\,z)^{-T}\Big) & \text{sinon.}\end{cases}$$

Lorsque l'opérateur D admet une singularité finie non nulle, on obtient ainsi des bornes fines. ♢

Démonstration. On traite séparément chaque classe de congruence modulo 1 des exposants λ_i. Pour une série généralisée dans laquelle tous les exposants de z qui apparaissent sont congrus modulo 1, on est dans la situation considérée ci-dessus. D'après le lemme 5.25, le lemme 6.19 est applicable avec $b(z) = M\,(1-\alpha\,z)^{-T-1}$, et il fournit des séries majorantes de la forme cherchée. La finesse pour un rayon de convergence fini résulte de ce que l'on peut alors choisir α égal à l'inverse du module de la singularité dominante de D, exactement comme en §5.3.3. □

Hélas, l'application aux séries généralisées du procédé de normalisation de la section 5.2.3 présente une difficulté technique non résolue à ce jour. Nos bornes au voisinage des points singuliers réguliers ne sont donc pas fines dans le cas d'un développement de rayon de convergence infini, non réduit à une série entière. Il est probablement possible de contourner cette difficulté en bornant les coefficients des séries φ_k de (6.10) de proche en proche, pour k décroissant, à partir du système de récurrences (4.24) page 74. Une adaptation de la méthode directe ci-dessus serait tout de même désirable.

Notons également, toujours comme piste d'amélioration future, que les bornes de cette section sont insatisfaisantes pour $|\lambda|$ grand.

Chapitre 7

Algorithmes de base et scindage binaire

Ce chapitre et le suivant sont centrés sur l'évaluation *ponctuelle* à *grande précision* des fonctions D-finies. Celui-ci commence par des rappels sur l'arithmétique à précision arbitraire avec des objets comme les entiers et les polynômes, qui serviront aussi, plus anecdotiquement, au chapitre 9. Nous étudions ensuite le calcul d'un terme d'une suite P-récursive par *scindage binaire*, à la base de la méthode d'évaluation des fonctions D-finies détaillée dans le chapitre suivant.

Une grande partie du contenu se trouve résumée dans un article présenté à ISSAC 2010 [174], et des versions préliminaires de certains points sont déjà dans mon rapport de M2 [173].

7.1 Introduction

Enfin un peu d'algorithmique ! Nous avons vu au chapitre 1 que les procédures d'évaluation numérique à grande précision offertes par NumGfun se fondent sur une technique appelée *scindage binaire* (en anglais *binary splitting*). Dans la littérature, ce nom fait référence au calcul par un algorithme « diviser pour régner » de la somme d'une série à coefficients rationnels [122, 134, 115, 40]. Le procédé s'applique classiquement aux séries hypergéométriques, et permet de calculer la n-ième somme partielle d'une telle série en temps quasi-linéaire $\tilde{O}(n)$. Qui plus est, il est réputé efficace en pratique [260, 49, 9].

De même que beaucoup d'algorithmes apparentés, il se réinterprète comme le calcul d'un *arbre de produits* de matrices à coefficients entiers [112, 57, 24]. Dans ce contexte, le calcul d'une somme partielle de série hypergéométrique n'est qu'un cas particulier de celui d'un terme d'une suite récurrente linéaire à coefficients polynomiaux. (La suite des sommes partielles d'une série D-finie, et en particulier d'une série hypergéométrique, est P-récursive d'après le corollaire 3.15.) Un arbre de produits convenable permet d'obtenir le n-ième terme d'une suite P-récursive quelconque sans calculer tous les précédents, et en temps quasi-linéaire en n. L'algorithme général est dû à Chudnovsky et Chudnovsky [56, 57]. L'histoire de l'idée, plus ancienne, est par exemple retracée par Bernstein [24, §12.7].

Ce chapitre est consacré principalement au calcul du n-ième terme d'une suite P-récursive par scindage binaire. On peut le voir comme la plus primitive des fonctionnalités d'évaluation numériques disponibles dans NumGfun : le calcul non pas de la valeur en un point d'une fonction D-finie, mais d'un des coefficients ou d'une des sommes partielles de son développement en série à l'origine. L'algorithme a été étudié à de multiples reprises dans toutes sortes de cas particuliers, témoins les articles cités ci-dessus et les références qu'ils mentionnent. Je m'attache ici à rassembler et adapter à la situation (plutôt générale) de cette thèse un certain nombre d'observations dispersées dans la littérature.

La section 7.2 présente des rappels sur la complexité algorithmique des opérations sur les entiers à précision arbitraire et de quelques autres opérations de base. Ceux-ci établis, nous nous intéressons (section 7.3) à l'algorithme de scindage binaire de Chudnovsky et Chudnovsky, qui est à la base des méthodes d'évaluation à grande précision des fonctions D-finies du chapitre suivant. Enfin, la section 7.4 recense des « astuces » algorithmiques qui permettent de gagner en efficacité par rapport à la description de la section 7.3. On aboutit également à une estimation plus fine, « sans $O(\cdot)$ », de la complexité du calcul des arbres de produits qui nous intéressent.

7.2 Complexité des opérations de base

7.2.1 Conventions

Notations. Dans ce chapitre et dans le suivant, on note $\mathbb{K} = \mathbb{Q}(i)$ et $\mathbb{A} = \mathbb{Z}[i]$. Par fidélité à l'implémentation et pour simplifier quelques évaluations fines de complexité, la plupart des algorithmes sont énoncés uniquement dans ce cadre. Ils se généralisent sans difficulté à la situation où $\mathbb{K}$ est un corps de nombres quelconque (plongé dans $\mathbb{C}$) et $\mathbb{A}$ son anneau des entiers.

Représentation des nombres et hauteur. On appelle *hauteur* d'un objet à coefficients entiers la taille maximale, en bits, des entiers qui apparaissent dans sa représentation. La hauteur du rationnel $\pm p/q$ est par définition

$$\mathrm{haut}(p/q) = \max\left(\lceil \lg p \rceil, \lceil \lg q \rceil\right),$$

celle d'un complexe $\frac{1}{d}(x + i\,y) \in \mathbb{K}$ ($x, y \in \mathbb{Z}$, $d \in \mathbb{N}$) est le maximum des hauteurs de x, y, d. Nous supposons plus généralement les éléments d'un corps de nombres fixé $\mathbb{Q}(\zeta)$ représentés sous la forme $\frac{1}{d}\sum_i x_i \zeta^i$ avec $x_i \in \mathbb{Z}$ et $d \in \mathbb{N}$; la hauteur d'un tel nombre algébrique est donc

$$\mathrm{haut}\left(\frac{1}{d}\sum_i x_i \zeta^i\right) = \max\left(h(d), h(x_0), h(x_1), \ldots\right).$$

La hauteur d'un polynôme, d'une matrice, d'une série à coefficients rationnels ou algébriques, ou d'une combinaison de tout cela est le maximum des hauteurs de ses coefficients.

Modèle de complexité. Sauf mention contraire explicite, les estimations de complexité des algorithmes font référence à la complexité *binaire* dans le *cas le pire*. Le qualificatif *binaire* signifie que les opérations élémentaires que l'on compte s'appliquent à un alphabet fini, canoniquement $\{0, 1\}$. En particulier, les opérations sur les entiers ou les flottants ne sont pas considérées comme élémentaires, mais ont un coût qui dépend de la taille des opérandes. Nous pourrons néanmoins toujours négliger la contribution des structures de contrôle et des « petits » entiers comme les indices de boucles devant celle des « grands » entiers qui apparaissent comme coefficients des données objet du calcul.

Complexité binaire s'oppose à *complexité arithmétique*. Dans une estimation de complexité arithmétique, on compte comme opérations de coût unitaire certaines opérations dans une structure algébrique donnée, par exemple les additions et multiplications dans l'anneau des coefficients d'un anneau de polynômes. À nouveau quoique pour une raison différente, le coût du contrôle est donc négligé.

Les estimations de complexité sont dans le *cas le pire*. Une borne supérieure de complexité est donc valable pour toute instance du problème. Une borne inférieure, par exemple celle sous-entendue dans une complexité $\Omega(f(n))$, doit seulement être réalisée par au moins une instance pour chaque valeur de n. Dans la mesure où les algorithmes dont nous étudions la complexité portent sur des objets denses, le cas le pire est en général aussi le cas *typique*, c'est-à-dire que l'ordre de grandeur de la complexité du pire cas est en fait atteint par « presque toute » instance, en un sens convenable. Nous recherchons au premier chef des algorithmes de complexité *quasi-optimale*. Formellement, nous dirons qu'une fonction f a une croissance quasi-linéaire si $f(n) = \tilde{O}(n) = O(n \lg^{O(1)} n)$, et que la complexité d'un algorithme est quasi-optimale si elle a une croissance quasi-linéaire en les tailles cumulées de l'entrée et de la sortie.

7.2.2 Arithmétique des grands entiers

L'efficacité des algorithmes considérés dans ce chapitre repose de façon cruciale sur la multiplication rapide de grands entiers. Le calcul sur les entiers, flottants, rationnels... de taille arbitraire, par opposition à leurs analogues confinés à quelques mots

machine, s'appelle arithmétique multiprécision. On rappelle ici quelques résultats bien connus sur le coût des opérations arithmétiques qui sont à la base des calculs. Tous les résultats énoncés sans référence sont détaillés dans le livre récent de Brent et Zimmermann [40], certains aussi dans celui de Crandall et Pomerance [73, Chap. 9].

Multiplication d'entiers. Par l'algorithme naïf « de l'école primaire », la multiplication de deux entiers de n bits prend $\Theta(n^2)$ opérations. Plusieurs algorithmes de *multiplication rapide* conduisent à de meilleures complexités. Bernstein [23] en présente un panorama concis et très complet. Ils s'organisent grossièrement en deux classes.

1. L'algorithme de Karatsuba et ses généralisations (schémas de Toom-Cook) sont des méthodes diviser-pour-régner simples. L'algorithme de Karatsuba repose sur la formule
$$(a\beta+b)\,(c\,\beta+d) = a\,c\,\beta^2 + \big((a+b)\,(c+d) - a\,c - b\,d\big)\beta + b\,d, \tag{7.1}$$
qui ramène la multiplication de deux entiers de taille n à trois multiplications en taille $n/2+O(1)$. Appliquée récursivement à profondeur $\lg_2 n$, elle conduit au final à effectuer $\Theta(n^{\log_2 3}) = O(n^{1,59})$ multiplications en taille $O(1)$. Les applications récursives de la relation (7.1) pour transformer les entiers initiaux puis recombiner les résultats des multiplications élémentaires prennent elles aussi un temps $\Theta(n^{\log_2 3})$.
2. Les algorithmes à base de transformée de Fourier rapide (FFT) conduisent quant à eux à des complexités quasi-linéaires. Ils ramènent la multiplication d'entiers de taille n à seulement $\Theta(n)$ multiplications d'entiers de taille bornée. Le passage des opérandes donnés en entrée aux petits entiers à multiplier au final et inversement se fait par des combinaisons de transformées de Fourier directes ou inverses dont les détails diffèrent suivant les algorithmes. Il demande $\Theta(n\,\lambda(n))$ opérations où $\lg n \leqslant \lambda(n) = O((\lg n)^{O(1)})$. Observons qu'à la différence de ce qui se passe avec les algorithmes de la première classe, le coût des produits ponctuels après transformée est négligeable devant celui de la transformée.

La meilleure complexité à ce jour est la suivante. Formellement, ce résultat se place dans le modèle de complexité « standard » des machines de Turing à plusieurs bandes [218]. Des complexités plus faibles sont possibles dans d'autres modèles, peut-être moins réalistes pour des entiers de très grande taille [145, p. 311].

Théorème 7.1. (Fürer [104]) On peut multiplier deux entiers de taille n en
$$O(n\,(\lg n)\,2^{O(\lg^* n)})$$
opérations binaires, où $\lg^*$ désigne le logarithme itéré, défini par $\lg^* n = 0$ pour $n \leqslant 1$ et $\lg^* n = \lg^*(\lg n) + 1$ sinon. ♢

En pratique, les implémentations efficaces de la multiplication de grands entiers sont des procédures récursives qui combinent plusieurs algorithmes en sélectionnant le plus rapide à chaque étape suivant la taille des opérandes. Il n'existe à ma connaissance pas d'implémentation de l'algorithme de Fürer. Cependant, d'autres algorithmes de multiplication par FFT sont largement utilisés. GMP [116], la bibliothèque d'arithmétique multiprécision la plus populaire, emploie une variante de l'algorithme de Schönhage-Strassen, de complexité $O(n \lg n \lg\lg n)$ (la meilleure connue avant le résultat de Fürer) pour multiplier des entiers dont la taille dépasse quelques dizaines de kilo-octets [111]. C'est sur GMP que repose l'arithmétique entière des versions récentes de Maple[1], utilisée de manière intensive par l'implémentation dans NumGfun des algorithmes de ce chapitre.

1. Précisément, Maple 15 (avril 2011) vient avec GMP 4.2.1 (mai 2006).

Fonction de multiplication. Il est d'usage d'exprimer le coût des opérations sur les grands entiers en fonction de celui de la multiplication. On impose en outre une poignée de contraintes techniques qui permettent de simplifier ces expressions.

DÉFINITION 7.2. On appelle fonction de multiplication une fonction $M\colon \mathbb{N} \to \mathbb{N}$ satisfaisant les propriétés (i) à (iii) suivantes.

(i) On peut multiplier deux entiers quelconques d'au plus n bits en au plus $M(n)$ opérations binaires.
(ii) La fonction $n \mapsto M(n)/n$ est croissante.
(iii) Pour tous $m, n \geqslant 1$, on a $M(mn) \leqslant m^2 M(n)$.

Dans toute la suite, M désigne une fonction de multiplication. ♢

Les hypothèses (ii) et (iii) de la définition 7.2 sont relativement standard [249, §8.3]. Le point (ii) entraîne en particulier

$$M(n) + M(m) \leqslant n \frac{M(m+n)}{m+n} + m \frac{M(m+n)}{m+n} = M(n+m).$$

Pour multiplier deux entiers déséquilibrés, c'est-à-dire de tailles m et $n > m$ significativement différentes, une stratégie simple consiste à se ramener à $\lceil n/m \rceil$ multiplications en taille m en découpant en blocs l'opérande le plus grand. On bornera donc le coût de cette opération par $\lceil n/m \rceil M(m)$.

D'après ce qui précède, on peut prendre $M(n) = \Theta(n \lg n \lg\lg n)$, et cela reflète le comportement pratique de la multiplication de très grands entiers. Pour des tailles modérées, $M(n) = \Theta(n^\alpha)$ avec $1 < \alpha \leqslant 2$ peut être plus réaliste.

Applications de la multiplication rapide. La multiplication rapide sert de brique de base à des algorithmes de complexité quasi-linéaire pour les autres opérations arithmétiques élémentaires sur les entiers, ainsi que sur les flottants multiprécision. Nous utiliserons les résultats suivants. L'article de synthèse de Bernstein [23] contient de nombreux autres exemples.

PROPOSITION 7.3. Soient p et q des entiers de taille en bits bornée par n. On peut calculer en $O(M(n))$ opérations :

(a) le quotient et le reste de la division euclidienne de p par q ;
(b) une approximation flottante à précision 2^{-n} de p/q. ♢

PROPOSITION 7.4. Soient p, q des entiers d'au plus n bits. On peut calculer le plus grand commun diviseur $p \wedge q$ de p et q ainsi que des cofacteurs u, v tels que

$$p \wedge q = u\,p + v\,q$$

en $O(M(n) \lg n)$ opérations. ♢

À nouveau, la bibliothèque GMP (à partir de la version 4.3.0 pour le pgcd, et de la version 5.0.0 concernant la division) fournit des implémentations qui atteignent ces complexités.

7.2.3 Polynômes

Les algorithmes rapides mentionnés plus haut pour les opérations de base sur les entiers admettent des analogues — en général plus simples — s'appliquant aux polynômes, éventuellement avec des restrictions sur l'anneau des coefficients. La complexité binaire est alors remplacée par la complexité arithmétique.

Pour tout anneau commutatif A où 2 est inversible, l'analogue polynomial de l'algorithme de Schönhage-Strassen permet de multiplier deux polynômes de $A[x]$ de degré borné par n en $O(n \lg n \lg\lg n)$ opérations dans A. L'algorithme de Cantor-Kaltofen [45] fonctionne sur tout anneau de base et aboutit à la même borne de complexité. Aucune adaptation polynomiale aussi générale de l'algorithme de Fürer n'est connue à ce jour.

On note aussi $M(n)$ une borne sur la complexité de la multiplication de polynômes. La définition est la même que la définition 7.2, en remplaçant l'hypothèse (i) par

(i') On peut multiplier deux polynômes de degrés bornés par n en au plus $M(n)$ opérations arithmétiques.

Le contexte indique en général laquelle des deux fonctions chaque occurrence de la notation $M(n)$ désigne ; au besoin, on écrira $M_{\mathbb{Z}}(n)$ ou $M_{A[x]}(n)$.

Ces résultats de complexité algébrique ne tiennent pas compte de la taille des coefficients. Dans le cas des polynômes à coefficients entiers, on a le résultat plus précis suivant.

Proposition 7.5. On peut calculer le produit de polynômes de degré d à coefficients entiers d'au plus h bits en $O(M_{\mathbb{Z}}(d\,h))$ opérations binaires. ♢

On en déduit une borne de complexité de $O(M_{\mathbb{Z}}(d^2\,h))$ pour le produit de polynômes de degré d et hauteur h à coefficients rationnels en utilisant le lemme 7.10 ci-dessous. Cette complexité reflète la taille du résultat dans le cas le pire.

7.2.4 Algèbre linéaire

Un algorithme *bilinéaire* [44, 5] pour le produit de matrices A et B à coefficients dans K est un algorithme qui calcule $A\,B$ en effectuant successivement

(i) des combinaisons linéaires s_i d'entrées de A et t_i d'entrées de B, à coefficients dans K ;

(ii) les produits $p_i = s_i\,t_i$;

(iii) puis des combinaisons linéaires des p_i donnant les entrées de $A\,B$.

On appelle exposant de l'algèbre linéaire sur un corps K l'infimum $\omega_{\text{inf}}(K)$ des ω tels que le produit de matrices $s \times s$ à coefficients dans K soit faisable en $O(s^\omega)$ opérations arithmétiques dans K par un algorithme bilinéaire. Il est connu que $\omega_{\text{inf}}(K)$ ne dépend que de la caractéristique de K [44, Chap. 15]. Les meilleures bornes sont actuellement $2 \leqslant \omega_{\text{inf}}(K) < 2{,}376$ [69], indépendamment de K.

L'algorithme de Coppersmith-Winograd duquel découle la borne supérieure est un exemple fameux d'algorithme « galactique » [160] — asymptotiquement efficace, mais au-delà d'un seuil qu'un calcul à l'échelle d'une galaxie ne suffit pas à atteindre. Pour les petites tailles qui nous intéressent, $\omega = 3$ est le plus réaliste. Un peu au-delà, la complexité de l'algorithme de Strassen, soit $\omega = \log_2 7$, est reflétée dans les expériences. Il faut aussi noter que pour des raisons liées à l'architecture des ordinateurs (et à l'effort consacré à développer des bibliothèques performantes !), l'algèbre linéaire sur certains domaines, à commencer par les flottants machine, est extrêmement efficace en pratique bien que les implémentations soient souvent de complexité cubique.

Définition 7.6. Dans toute la suite, on note ω un réel tel que le produit de matrices $s \times s$ à coefficients dans un corps K quelconque puisse être effectué en $O(s^\omega)$ opérations arithmétiques dans K par un algorithme bilinéaire. ♢

Comme dans le cas des entiers et des polynômes, la multiplication de matrices sert d'étalon de complexité pour les opérations d'algèbre linéaire.

Théorème 7.7. Soit K un corps, et supposons que l'on dispose d'un algorithme pour multiplier les matrices de $K^{s\times s}$ en $O(n^\omega)$ opérations dans K. Alors on peut calculer

- le déterminant, le rang ou une base du noyau d'une matrice de $K^{s\times s}$;
- l'inverse d'une matrice de $\mathrm{GL}_s(K)$

en $O(s^\omega)$ opérations. ♢

Je renvoie à [44, Chap. 15] ou encore [33, chap. 3] pour des preuves et références. Pour les matrices à coefficients entiers, une approche relativement naïve aboutit aux complexités suivantes, qui suffisent à nos besoins. Des résultats plus fins existent pour l'inversion et bien d'autres problèmes apparentés [248, 227, 228].

THÉORÈME 7.8. Soient $A, B \in \mathbb{Z}^{s\times s}$ de hauteur bornée par h.

(a) On peut calculer le produit $A\,B$ en $O(s^\omega\, M(h+\lg s))$ opérations binaires.

(b) On peut calculer $\det A \in \mathbb{Z}$ et $\det(A)\,A^{-1} \in \mathbb{Z}^{s\times s}$ en

$$O\big(M\big(s^3\,(\lg s)^{O(1)}\,h\lg h\big) + s^{\omega+1}\,(\lg s)^{O(1)}\,h\big) = O\big(s^{\omega+1+o(1)}\,M(h)\lg h\big)$$

opérations binaires. ♢

DÉMONSTRATION (ESQUISSE). (a) La taille des entiers manipulés dans les étapes intermédiaires d'un algorithme bilinéaire de multiplication est bornée par $h + 2\lg s + O(1)$. (b) Le déterminant et donc l'inverse de A sont de hauteur $O(s\,(h+\lg s))$. On en déduit la complexité par des techniques standard de calcul multimodulaire. □

7.3 Calcul d'un terme d'une suite P-récursive

7.3.1 Scindage binaire

Considérons une relation de récurrence linéaire de la forme

$$u_{n+s} + b_{s-1}(n)\,u_{n+s-1} + \cdots + b_0(n)\,u_n = 0, \quad b_k \in \mathbb{K}(n) = \mathbb{Q}[i](n). \tag{7.2}$$

On voit facilement que $\text{haut}(u_n) = O(n \lg n)$; et, pour un choix de b_k et de conditions initiales suffisamment général, cette borne est optimale. Calculer successivement $u_0, \ldots, u_{N-1}$ en « déroulant » la récurrence (7.2) demande donc

$$\Theta(N^2\, M(\lg N))$$

opérations binaires. Cette complexité est quasi-linéaire en la taille combinée des N premiers termes de u, mais quadratique en celle de u_N.

La méthode dite de *scindage binaire* est un algorithme « diviser pour régner » qui calcule un seul terme u_N en complexité quasi-optimale. On commence par réécrire (7.2) comme une récurrence matricielle d'ordre 1 :

$$\begin{pmatrix} u_{n+1} \\ u_{n+2} \\ \vdots \\ u_{n+s} \end{pmatrix} = \begin{pmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ -b_0(n) & -b_1(n) & \cdots & -b_{s-1}(n) \end{pmatrix} \begin{pmatrix} u_n \\ u_{n+1} \\ \vdots \\ u_{n+s-1} \end{pmatrix}.$$

On multiplie cette récurrence par un dénominateur commun $q(n) \in \mathbb{Z}[n]$ des entrées de la matrice, ce qui donne

$$q(n)\,U_{n+1} = B(n)\,U_n \tag{7.3}$$

où

$$U_n = \begin{pmatrix} u_n \\ u_{n+1} \\ \vdots \\ u_{n+s-1} \end{pmatrix}, \qquad B(n) = q(n) \begin{pmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ -b_0(n) & -b_1(n) & \cdots & -b_{s-1}(n) \end{pmatrix} \in \mathbb{A}[n].$$

Posons $P(i,j) = B(j-1)\cdots B(i+1)\,B(i)$. On a ainsi

$$U_N = \frac{1}{\prod_{i=0}^{N-1} q(i)}\, P(0,N).$$

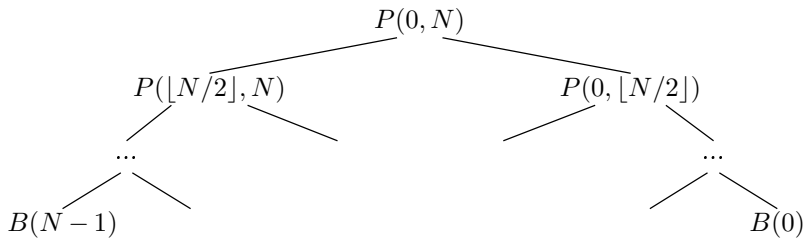
On calcule $P(0,N)$ en appliquant récursivement la formule

$$P(i,j) = P\Big(\Big\lfloor \frac{i+j}{2} \Big\rfloor, j\Big)\, P\Big(i, \Big\lfloor \frac{i+j}{2} \Big\rfloor\Big), \tag{7.4}$$

et l'on procède de même, mais séparément, pour obtenir le dénominateur $\prod_{i=0}^{N-1} q(i)$.

Le fait de calculer séparément numérateur et dénominateur signifie en fait que l'on s'abstient de simplifier les fractions qui apparaissent au cours du calcul. C'est parce que le coût d'un pgcd d'entiers de taille n est plus élevé d'un facteur $\Theta(\lg n)$ que celui d'une multiplication. Sauf situation exceptionnelle, retirer les facteurs communs ne diminue pas suffisamment les tailles des numérateurs et dénominateurs pour compenser cet écart ; et au final, l'algorithme sans simplifications est plus rapide d'un facteur $\Theta(\lg n)$.

La structure



construite par la règle (7.4) s'appelle un *arbre de (sous-)produits*. Les arbres de produits, d'entiers et de polynômes notamment, ont de nombreuses applications en calcul formel et dans d'autres domaines de l'informatique [24, §12-§16, §18, §23]. Leur emploi pour tirer le meilleur parti de la multiplication rapide en équilibrant les opérandes date au moins des années 1970 [24, §12.7].

7.3.2 Complexité

Les deux énoncés suivants reprennent essentiellement la formulation de l'algorithme de scindage binaire pour les suites P-récursives quelconques due à Chudnovsky et Chudnovsky [57, §2]. Une petite complication vient de ce qu'ils omettent apparemment de prendre en compte le coût de l'évaluation des polynômes qui apparaissent aux feuilles de l'arbre de produits. Or, en arithmétique multiprécision, l'évaluation d'un polynôme par l'algorithme de Horner a un coût quadratique en le degré, qui ne rentre pas dans la borne de complexité annoncée.

Proposition 7.9. [57] Supposons $B(n) \in \mathbb{A}[n]^{s\times s}$ de degrés bornés par d' et hauteurs bornées par h'. Le calcul par scindage binaire du produit

$$P(0,N) = B(N-1)\cdots B(1)\,B(0) \in \mathbb{K}^{s\times s} \qquad (N \geqslant s)$$

demande $O\Big(s^\omega\, M\big(N\,(h'+d'\lg N)\big)\lg N\Big)$ opérations binaires pour $d' = O(N)$. ♢

Démonstration. Posons $H = h' + d'\lg N$. Pour $n < N$ fixé, la hauteur de $B(n)$ est $O(H)$, donc celle de $P(i,j)$ est $O\big((j-i)\,H\big)$, et la multiplication (7.4) prend

$$O\big(s^\omega\, M\big((j-i)\,H\big)\big)$$

opérations binaires. Comme $M(m)+M(n) \leqslant M(m+n)$ d'après la définition 7.2, le coût total des multiplications pour obtenir les éléments à profondeur k de l'arbre des produits à partir de ceux à profondeur $k+1$ est $O(s^\omega\, M(N\,H))$, indépendamment de k. Une fois les $B(n)$ pour $0 \leqslant n < N$ connus, le calcul de $P(0,N)$ demande donc $O(s^\omega\, M(NH)\lg N)$ opérations.

Le calcul de chacune des N matrices aux feuilles de l'arbre demande d'évaluer $O(s^2)$ polynômes de degré au plus d' et de hauteur au plus h' en un point n de hauteur bornée par $\lceil \lg N \rceil$. Pour cela, on utilise à nouveau un algorithme « diviser pour régner » : on calcule $p(n)$ et $n^{\deg p}$ par les formules[2]

$$\begin{cases} p(n) = n^{\lfloor \deg p/2 \rfloor}\, p_{\text{haut}}(n) + p_{\text{bas}}(n), \quad \deg p_{\text{haut}} = \lceil \tfrac{1}{2}\deg p \rceil, \deg p_{\text{bas}} = \lfloor \tfrac{1}{2}\deg p \rfloor \\ n^{\deg p} = \big(n^{\lfloor \deg p/2 \rfloor}\big)^2\, n^{(\deg p) \bmod 2} \end{cases}$$

en évaluant récursivement p_{haut}, p_{bas} et $n^{\deg p}$. La complexité est de $O(M(H)\lg d')$ opérations par polynôme, soit $O(s^2 N M(H)\lg d')$ au total. □

Pour appliquer la proposition précédente aux récurrences scalaires, il reste à contrôler la complexité de la conversion de l'entrée (7.2) sous la forme (7.3). Dans le pire des cas, la croissance des coefficients induite par les besoins de dénominateurs communs à différentes étapes fait croître le coût total du calcul d'un facteur de l'ordre de $s\,d$, où d désigne le degré des coefficients de (7.2). On peut cependant bien souvent borner plus finement le degré d' et la hauteur h' qui interviennent dans le proposition précédente, par exemple dans la situation commune où tous les coefficients de (7.2) ont le même dénominateur. Nous énonçons donc la complexité de l'algorithme complet à la fois en termes de (h,d) et de (h',d').

LEMME 7.10. Soit $p\in\mathbb{K}[x]$ un polynôme de degré d et de hauteur h. On peut réduire au même dénominateur les coefficients de p, c'est-à-dire calculer $d'\in\mathbb{N}$ et $p'\in\mathbb{A}[x]$ tels que $p(x)=\frac{1}{d'}\,p'(x)$, en $O(M(d^2\,h)\lg d)$ opérations. Le numérateur p' obtenu est de hauteur bornée par $(d+1)\,h$. ♢

DÉMONSTRATION. Soient $q_0,\dots,q_d$ les dénominateurs des coefficients de p. On construit l'arbre des sous-produits de $q_0\cdots q_d$ en $O(M(d\,h)\,\lg\,d)$ opérations, et l'on en déduit chaque $q_0\cdots\hat{q_i}\cdots q_d$ en temps $O(M(d\,h))$ par produits successifs le long d'une branche de l'arbre, ou encore en divisant $q_0\cdots q_d$ par q_i. □

PROPOSITION 7.11. [57] Soit u une suite P-récursive sur $\mathbb{K}$ donnée par une récurrence de la forme (7.2) et des conditions initiales elles-mêmes dans $\mathbb{K}$. Supposons que les coefficients $b_k(n)$ de (7.2) n'ont pas de pôles entiers positifs. Soient d une borne sur les degrés de leurs numérateurs et dénominateurs, h une borne sur leurs hauteurs et celles des conditions initiales. Soient d',h' des bornes sur les degrés et hauteurs des coefficients $B(n)$ et $q(n)$ de (7.3) correspondants. L'algorithme de scindage binaire calcule le terme $u_N\in\mathbb{K}$ $(N\geqslant s)$ de u en

$$O\Big(s^\omega\,M\big(N\,(h'+d'\lg N)\lg(N\,h')\big)\Big)=O\Big(s^\omega\,M\big(s\,d\,N\,(h+\lg N)\big)\lg(N\,h)\Big)$$

opérations binaires si $s,d=O(\lg N)$. Sous les mêmes hypothèses, le coût descend à

$$O\Big(s^\omega\,M\big(N\,(h'+d'\lg N)\lg N\big)\Big)=O\Big(s^\omega\,M\big(s\,d\,N\,(h+\lg N)\big)\lg N\Big),$$

si l'on ne demande pas que le résultat soit sous forme irréductible. ♢

DÉMONSTRATION. Pour passer de (7.2) à (7.3), on commence par écrire chacun des b_k comme un quotient de polynômes à coefficients dans $\mathbb{A}$. Le résultat est de hauteur $h''\leqslant 2\,(d+1)\,h$, et cette étape demande $O(s\,M(d\,h)\lg d)$ opérations. (Le facteur 2 provient de la multiplication du numérateur par le dénominateur commun des coefficients du dénominateur, et inversement.) Il reste alors à réduire au même dénominateur (polynomial cette fois) les s coefficients non constants de l'équation (7.2). Le résultat est de degré $d'\leqslant s\,d$ et de hauteur

$$h'\leqslant s\,(h''+\lceil\lg s\rceil+\lceil\lg d\rceil)=s\,(h''+o(\lg N)).$$

Cette réécriture a un coût de $O(M(s\;d'\,h')\;\lg\;s)$ opérations, par l'argument du lemme 7.10 où l'on a remplacé la multiplication d'entiers de taille h par celle de polynômes de degré d' et hauteur h'.

On calcule ensuite le produit $P(0,N)$ en $O(s^\omega\,M(N\,H)\lg N)$ opérations, où

$$H=h'+d'\lg N\leqslant s\,d\,(2\,h+\lg N)\,(1+o(1)),$$

2. Cet algorithme a été proposé dans un contexte différent par Estrin [88]. Bostan *et. al.* [34, §3.3] relèvent son intérêt pour évaluer un polynôme en temps quasi-linéaire en son degré quand les coefficients sont des entiers multiprécision.

d'après la proposition 7.9. Le calcul du dénominateur $\prod_n q(n)$ demande quant à lui $O(M(N\,H)\lg N)$ opérations. Appliquer la matrice obtenue aux conditions initiales en prend $O(s^2\,M(N\,H))$. Sous l'hypothèse que $s, d = O(\lg N)$, le coût de toutes les autres étapes à ce stade est négligeable devant celui du calcul de $P(0, N)$.

Enfin, la mise sous forme irréductible du résultat consiste en un pgcd d'entiers de taille $O(NH)$, faisable en $O(M(NH)\lg(NH))$ opérations binaires. □

Comme la hauteur de u_N peut atteindre $\Omega((N+h)\lg N)$, ce résultat est optimal vis-à-vis de N et h, à des facteurs logarithmiques près.

Le même algorithme est applicable en remplaçant $\mathbb{K} = \mathbb{Q}(i)$ par un corps de nombres [235]. Cela est utile pour l'évaluation « aux singularités » (voir §8.6) des fonctions D-finies sur $\mathbb{Q}$. Plus généralement, on peut énoncer des résultats similaires pour le calcul d'arbres de produits dans une $\mathbb{Z}$- ou $\mathbb{Q}$-algèbre sans torsion arbitraire (dans ce dernier cas, on écrit $A = \mathbb{Q} \otimes_{\mathbb{Z}} A'$ et l'on effectue les multiplications dans $\mathbb{Z} \times A'$) [173]. La seule différence notable est que faute de choix d'une base de l'espace, la hauteur d'un élément n'est définie qu'à une constante additive près.

La section 7.4 en fin de chapitre discute différentes améliorations du facteur indépendant de N et h dans les estimations de complexité des deux propositions précédentes. Nous verrons que sous des hypothèses un peu plus fortes, le coût du calcul est de l'ordre de $\frac{1}{3}\,s^2\,M(NH)\lg N$.

7.3.3 Sommes de séries

Nombre de récurrences particulières peuvent être mises sous une forme matricielle qui présente une certaine structure préservée au fil des multiplications. Soit par exemple à calculer $\sigma_n = \sum_{k=0}^{n-1} u_k$, où $(u_n) \in \mathbb{K}^{\mathbb{N}}$ est donnée par la récurrence

$$u_{n+s} + b_{s-1}(n)\,u_{n+s-1} + \cdots + b_0(n)\,u_n = 0 \tag{7.5}$$

et des conditions initiales $u_0, \ldots, u_{s-1}$. On en déduit

$$(\sigma_{n+s+1} - \sigma_s) + b_{n+s-1}(n)\,(\sigma_{n+s} - \sigma_{n+s-1}) + \cdots + b_0(n)\,(\sigma_{n+1} - \sigma_n) = 0, \tag{7.6}$$

c'est-à-dire une récurrence d'ordre $s+1$ satisfaite par σ, que l'on peut écrire sous forme matricielle avant de poursuivre comme ci-dessus.

Cependant, il est préférable de « dérouler » simultanément (7.5) et la récurrence inhomogène $\sigma_{n+1} - \sigma_n = u_n$, sous la forme

$$\begin{pmatrix} u_{n+1} \\ \vdots \\ u_{n+s-1} \\ u_{n+s} \\ \sigma_{n+1} \end{pmatrix} = \begin{pmatrix} & 1 & & \mathbf{0} & 0 \\ & & \ddots & & \vdots \\ \mathbf{0} & & & 1 & 0 \\ * & * & \cdots & * & 0 \\ 1 & 0 & \cdots & 0 & 1 \end{pmatrix} \begin{pmatrix} u_n \\ \vdots \\ u_{n+s-2} \\ u_{n+s-1} \\ \sigma_n \end{pmatrix}. \tag{7.7}$$

En effet, le produit de deux matrices triangulaires par blocs est triangulaire par blocs. Lorsque l'on sépare numérateur et dénominateur de la matrice de (7.7) et que l'on forme l'arbre des produits correspondant au numérateur, les s coefficients supérieurs de la colonne de droite restent nuls pour tous les sous-produits intermédiaires, et le coefficient du bas n'est autre que le dénominateur. La même idée s'applique dès que l'opérateur de récurrence annulant la suite à calculer se factorise [173, §2.3].

Notons $\mathrm{MM}(s)$ la complexité multiplicative du produit de matrices $s \times s$ à coefficients entiers, c'est-à-dire le nombre de multiplications dans $\mathbb{Z}$ auxquelles se ramène une multiplication dans $\mathbb{Z}^{s\times s}$, sans compter les multiplications par des constantes. Chaque produit de matrices de la forme (7.7) représente $\mathrm{MM}(s) + s^2 + s + 1$ multiplications scalaires, contre $\mathrm{MM}(s+1) + 1$ pour la variante fondée sur la récurrence (7.6), où l'on calcule un produit de matrices pleines et un dénominateur séparé. La formule (7.7) est donc plus efficace dès que $\mathrm{MM}(s+1) - \mathrm{MM}(s) \geqslant s\,(s+1)$. Ce n'est pas le cas asymptotiquement avec $\mathrm{MM}(s) = O(s^\omega)$ et $\omega < 3$. En revanche, c'est vrai avec

l'algorithme naïf[3], de même — en anticipant sur la section 7.4 — qu'avec l'algorithme de Waksman ou avec les complexités du tableau 7.1 (p. 136) jusqu'à $s=16$.

EXEMPLE 7.12. Soit à calculer $e_n = \sum_{k=0}^{n-1} \frac{1}{k!}$. Le numérateur de la matrice de la récurrence (7.7) correspondante est $B(n) = \left(\begin{smallmatrix} 1 & 0 \\ n+1 & n+1 \end{smallmatrix}\right)$. On forme des produits partiels $B(j-1)\cdots B(i) = \left(\begin{smallmatrix} 1 & 0 \\ P(i,j) & Q(i,j) \end{smallmatrix}\right)$ par scindage binaire suivant la relation

$$\begin{pmatrix} 1 & 0 \\ P(i,j) & Q(i,j) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ P(m,j) & Q(m,j) \end{pmatrix} \begin{pmatrix} 1 & 0 \\ P(i,m) & Q(i,m) \end{pmatrix} \quad (m = \lfloor \tfrac{i+j}{2} \rfloor).$$

Cette relation est équivalente à la méthode classique de calcul de e_n par scindage binaire [115], qui consiste à appliquer récursivement les relations

$$P(i,j) = P(i,m)\,Q(m,j) + P(m,j), \quad Q(i,j) = Q(i,m)\,Q(m,j).$$

De même, pour une somme hypergéométrique générale (c'est-à-dire lorsque u_n satisfait une récurrence du premier ordre), le produit de matrices de la forme (7.7) requiert les quatre mêmes multiplications essentielles que la formulation classique [49]. Le coût de la méthode générale spécialisée à ce cas et celui des formules spécifiques sont donc les mêmes à un terme linéaire en n près. ♢

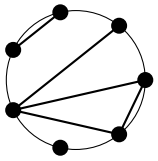
Des observations comparables s'appliquent notamment à la propagation des zéros dans les produits partiels de matrices qui donnent les coefficients et sommes partielles de séries paires impaires. Tout ceci milite en faveur de l'écriture (7.7) de la récurrence de préférence à des versions plus denses, conduisant à des matrices de plus petite taille, mais qui pourraient faire perdre une certaine structure.

7.3.4 Implémentation

NumGfun contient deux implémentations du calcul du N-ième terme d'une suite récurrente par scindage binaire. L'une est écrite en pur langage Maple, l'autre en C, au-dessus de la couche `mpz` de GMP, et fait partie de la bibliothèque `libNumGfun` mentionnée en §1.3. La version C est encore à un stade peu avancé. D'après mes premières expériences, elle est nettement plus efficace pour N modéré (voir figure 7.1), tandis que les deux implémentations présentent des performances comparables pour N assez grand[4], l'essentiel du temps étant alors passé dans GMP.

L'utilisateur dispose d'une fonction `NumGfun:-nth_term` qui prend en paramètre une suite P-récursive $u \in \mathbb{K}^{\mathbb{N}}$ et un indice N, et calcule u_N par l'algorithme de la section précédente. Par ailleurs, les procédures calculant u_n en fonction de n renvoyées par `gfun:-rectoproc` (voir §1.2.1) appellent automatiquement `nth_term` si le type d'évaluation demandée le permet et si N est suffisamment grand vis-à-vis de l'ordre de la récurrence. Enfin, la variante `fnth_term` calcule une approximation numérique de u_n sans passer par son expression exacte dans $\mathbb{Q}(i)$, ce qui économise un coûteux calcul de pgcd entre le numérateur et le dénominateur obtenus par scindage binaire. Les exemples suivants illustrent l'utilisation de ces fonctions pour calculer un terme d'une suite d'entiers ou approcher la limite d'une suite convergente. Comme d'habitude, les temps de calcul indiqués correspondent à la version Maple du code de scindage binaire.

EXEMPLE 7.13. Les nombres de Motzkin M_n (OEIS A001006 [220]) dénombrent les façons de relier n points répartis sur un cercle par des cordes qui ne s'intersectent pas. Ils satisfont la récurrence



$$(n+4)\,M_{n+2} = 3\,(n+1)\,M_n + (2\,n+5)\,M_{n+1}$$

3. Cette conclusion est en désaccord avec celle de Zimmermann [267, p. 25], qui ne tient pas compte de la forme particulière des matrices.

4. Avec Maple 11 ou ultérieur, les versions plus anciennes souffrant de problèmes d'efficacité dans les opérations sur les grands entiers.

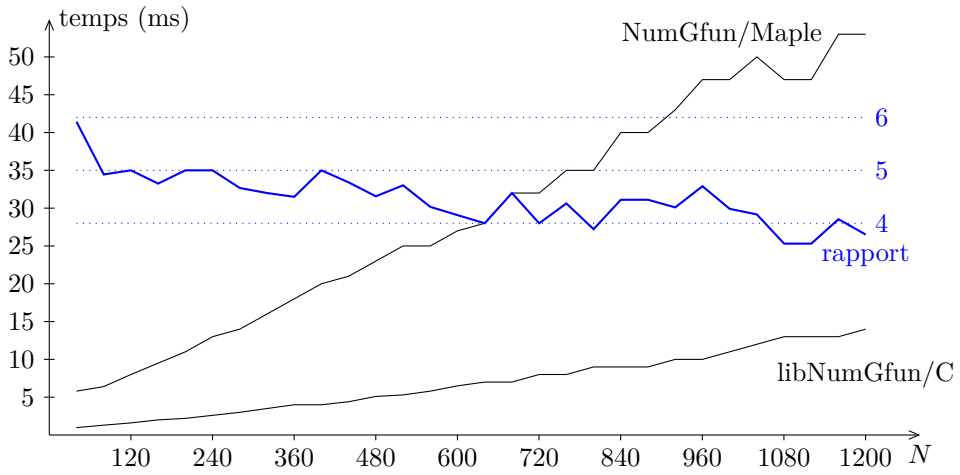


Figure 7.1. Temps de calcul comparés, par scindage binaire, des N-ièmes termes d'une base de solutions de la récurrence $(n^2+5)\,u_{n+3} = (n+1)\,(n+2)\,u_{n+2} + (n+2)\,u_{n+1} + u_n$ en pur Maple et avec un noyau en C utilisant GMP. Seule est mesurée la durée du calcul de l'arbre de produits, sans la division finale. Moyenne de $3 \leqslant k \leqslant 100$ exécutions, utilisant un seul cœur du processeur. L'écart se resserre quand N grandit : le rapport de performance entre les deux implémentations est inférieur à 2 pour $N = 100000$.

avec $M_0 = 1$, $M_1 = 1$. En utilisant NumGfun, on calcule

```
> nth_term({(n+4)*M(n+2)=3*(n+1)*M(n)+(2*n+5)*M(n+1), M(0)=1, M(1)=1},
          M(n), 100000);
```

6187829384...(47685 chiffres)...4866467713

en 3,1 secondes. De même, le calcul de

$$M_{10^6} = 2635090613...(477093\ \text{chiffres})...6434199151$$

prend 36,2 secondes. En comparaison, dérouler naïvement la récurrence demande 7,2 secondes pour M_{10^5} et 28,5 secondes pour $M_{2\cdot 10^5}$. Cet exemple n'est pas à l'avantage du scindage binaire, car toutes les divisions qui apparaissent au cours du calcul sont exactes, ce dont `nth_term` ne tire pas parti. ♢

Exemple 7.14. Par intégrations par parties successives à partir de la représentation intégrale de la fonction Gamma, on obtient

$$\Gamma(z) = \sum_{n=0}^{\infty} \frac{e^{-t}\,t^{n+z}}{z\,(z+1)\cdots(z+n)} + \int_t^{\infty} e^{-u}\,u^{z-1}\,\mathrm{d}u, \qquad 0 < \operatorname{Re} z < 1. \tag{7.8}$$

Brent [39, §6.10] a proposé d'utiliser cette expression pour évaluer $\Gamma(z)$ quand z est un rationnel de hauteur modérée. (La même approche s'étend immédiatement à des points algébriques, à nouveau de hauteur modérée.) Prenons $z = 1/3$. Le terme général u_n de la somme (7.8) satisfait alors $(3\,n+4)\,u_{n+1} = 3\,t\,u_n$. Avec $t = 29^3$, on vérifie que

$$\left| \sum_{n=65000}^{\infty} \frac{e^{-t}\,t^{n+z}}{z\,(z+1)\cdots(z+n)} + \int_t^{\infty} e^{-u}\,u^{z-1}\,\mathrm{d}u \right| \leqslant 10^{-10002}.$$

Ainsi, on détermine en 1,16 s l'approximation à 10^{-10000} près de $\Gamma(1/3)$

```
> a := 29: t := a^3: N := 65000:
> rec := {(3*n+4)*u(n+1) = 3*t*u(n), u(0) = 3*a}:
> evalf[10000](fnth_term(rec, u(n), N, 10002, 'series')
              * evalf[10002](exp(-t)));
```

2,6789385347...(9980 chiffres)...7239199978

(l'option `series` sert à sommer la série de terme général donné par la récurrence comme décrit en §7.3.3)

Le même calcul avec l'implémentation de la fonction Gamma fournie par Maple :

```
> evalf[10001](GAMMA(1/3))

   2,6789385347...(9980 chiffres)...7239199978
```

prend plus de 5 min. La comparaison n'est pas équitable, puisque l'implémentation de Maple, conçue pour des points flottants quelconques, ne tire pas parti de la petite taille du point d'évaluation. Elle met cependant en lumière l'importance de traiter spécifiquement les évaluations aux points rationnels simples si elles interviennent fréquemment, et notamment pour calculer les fonctions d'Airy à partir des conditions initiales

$$\mathrm{Ai}(0) = \frac{1}{3^{2/3}\,\Gamma(2/3)}, \quad \mathrm{Ai}'(0) = -\frac{3^{1/6}\,\Gamma(2/3)}{2\,\pi}, \quad \mathrm{Bi}(0) = \frac{1}{3^{1/6}\,\Gamma(2/3)}, \quad \mathrm{Bi}'(0) = \frac{3^{2/3}\,\Gamma(2/3)}{2\,\pi}$$

par l'algorithme présenté en §8.2. ♢

7.4 « Optimisation » du scindage binaire

Plusieurs techniques plus ou moins classiques permettent d'améliorer d'un facteur constant mais significatif la complexité du calcul d'un arbre de produits. Cette section, essentiellement bibliographique, vise à les passer en revue. Il n'y a pas d'idée nouvelle, et à de rares exceptions près, ces techniques ne sont pas actuellement utilisées dans NumGfun, le langage Maple s'y prêtant assez peu. J'ai en revanche tenté de rassembler les observations théoriques que devrait prendre en compte une implémentation « optimisée » des algorithmes de scindage binaire. Il s'agit donc en quelque sorte d'une liste de pistes pour le développement de `libNumGfun`.

Nous allons distinguer deux modèles de calcul. Dans le *modèle FFT*, on suppose les multiplications entières effectuées par transformée de Fourier discrète. On tire parti de la structure de l'algorithme de multiplication — transformée coûteuse, produits ponctuels bon marché, transformée inverse coûteuse — pour réorganiser le calcul de manière à économiser des transformées. Dans le *modèle boîte noire* au contraire, la multiplication est une opération indivisible, et les « optimisations » visent à limiter le nombre et la taille des multiplications. Ce second modèle offre moins d'opportunités d'accélération du code, mais reflète l'interface communément offerte par les implémentations de la multiplication d'entiers.

7.4.1 Modèle FFT

Opérations sur les transformées de Fourier. La technique la plus importante est diversement appelée *FFT caching* et *FFT addition* [24], *computing in the FFT mode* [235] ou encore *FFT invariance* [188]. On la trouve le plus souvent énoncée dans le contexte de la multiplication de polynômes dans $\mathbb{C}[x]$ ou plus généralement sur un anneau qui supporte la FFT. La multiplication de polynômes de degrés strictement inférieurs à d (que nous supposerons pour simplifier être une puissance de 2) se décompose alors en deux transformées de Fourier directes en taille $2\,d$, suivies de d multiplications dans l'anneau de base et d'une transformée inverse. Notons $F(n) = O(n \lg n)$ le coût d'une transformée de Fourier, directe ou inverse. Le coût de la multiplication dans $\mathbb{C}[x]$ est alors de $M(d) = 3\ F(2\ d) + O(d)$ opérations arithmétiques. Pour multiplier deux matrices $s \times s$ dont les entrées sont des polynômes de degrés comparables et tous bornés par d, on calcule une fois pour toutes les transformées directes de toutes les entrées, puis l'on effectue les produits deux à deux des évaluations aux racines $2\,d$-ièmes de l'unité des matrices de départ, et on en déduit le résultat par des transformées de Fourier inverse (voir figure 7.2). Le coût total de

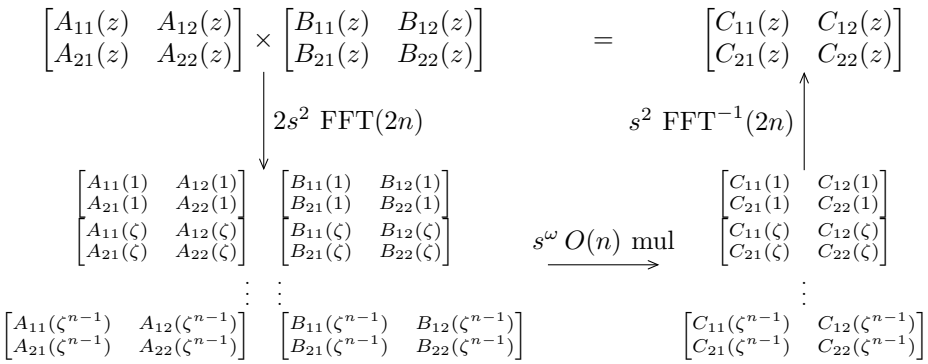


Figure 7.2. Produit de matrices de polynômes « par FFT ».

cet algorithme est de $3\,s^2\,F(2d) + O(s^\omega\, d) = s^2\,M(d) + O(s^\omega\, d)$ opérations dans $\mathbb{C}$. En comparaison, la méthode naïve consistant à calculer séparément chacun des produits dans $\mathbb{C}[x]$ intervenant dans la formule de multiplication matricielle demande $s^\omega\, M(d)$ opérations, soit $s + o(1)$ fois plus si $\omega = 3$ et $M(d)/d \gg s$.

De même, l'algorithme de Schönhage-Strassen réduit la multiplication dans $\mathbb{Z}/_{(2^n+1)\mathbb{Z}}$ à $O(\sqrt{n})$ multiplications dans $\mathbb{Z}/_{(2^m+1)\mathbb{Z}}$ avec $m = O(\sqrt{n})$, effectuées récursivement. Moyennant une profondeur de récursion de $\lg\lg n$, on se ramène ultimement à $O(n^{1-1/\lg n}) = O(n)$ multiplications en taille $O(n^{1/\lg n}) = O(1)$. On peut voir ces réductions successives comme une unique « transformée de Schönhage-Strassen »[5] qui ramène la multiplication d'entiers de taille n à $O(n)$ multiplications en taille $O(1)$, le tout en $F(n) = O(n \lg n \lg\lg n)$ opérations binaires. Cette transformée est un morphisme d'anneaux, et ramène donc aussi le produit de deux matrices $s \times s$ de hauteur h à $O(h)$ produits de matrices de hauteur $O(1)$. Le coût total de la multiplication est alors

$$3\,s^2 F(n) + O(s^\omega\, n) = s^2\,M(n) + O(s^\omega\, n).$$

La technique s'applique à l'identique à n'importe quelle autre $\mathbb{Z}$-algèbre.

Cette idée admet des variantes théoriquement un peu moins bonnes sur le plan de la complexité mais plus faciles à mettre en œuvre. En multipliant les entiers de taille n « comme des polynômes à coefficients réels de degré $n/\lg n$ », par une transformée de Fourier implémentée en arithmétique flottante à précision $O(\lg n)$ et sans appels récursifs (le cas de base étant réalisé par un algorithme de multiplication des flottants de complexité $M(n)$), on aboutit pour le produit matriciel à une complexité

$$O\left(s^2\, n\, M(\lg n) + s^\omega\, n\, \frac{M(\lg n)}{\lg n} \right).$$

Une seconde possibilité est de réduire la multiplication de matrices $A, B \in \mathbb{Z}^{s\times s}$ à celles de $A \bmod p$ par $B \bmod p$ pour un nombre suffisant de p premiers entre eux par réduction-reconstruction multimodulaire rapide [40, §2.7], la multiplication de matrices de hauteur n se fait alors en

$$O\left(s^2\, M(n) \lg n + s^\omega\, n\, \frac{M(\lg n)}{\lg n} \right)$$

opérations binaires. (Storjohann [226] détaille l'analyse, choix des p compris.) Le mérite de cette approche est qu'elle peut s'implémenter par-dessus une routine boîte noire de multiplication entière.

Pour des récurrences d'ordre élevé, on peut s'attendre à ce que les variantes les plus efficaces soient celles qui se ramènent à des multiplications de matrices de flottants machine, qui peuvent alors être effectuées à l'aide de BLAS[6] extrêmement

5. J'ai appris cette présentation de Joris van der Hoeven. La synthèse de Bernstein [23] adopte un point de vue analogue.

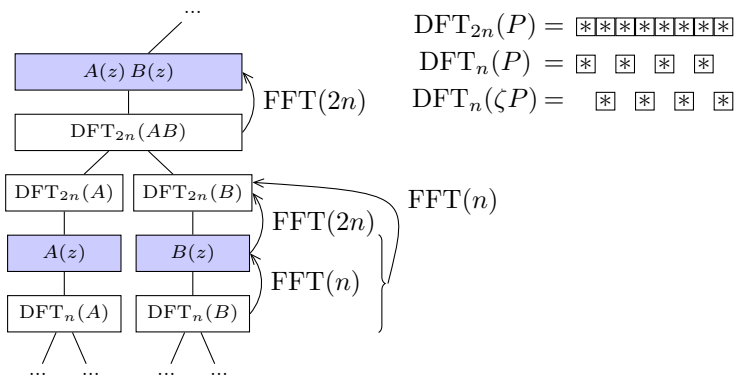


Figure 7.3. *FFT doubling* dans un arbre de produits de polynômes.

rapides comme ATLAS ou GOTOBlas [254, 113, 224]. On peut les combiner avec une bibliothèque de FFT comme FFTW [100] pour tirer le meilleur parti de l'effort de développement d'outils de calcul scientifique performants. Le prix à payer est qu'il devient difficile de garantir l'absence d'erreurs dues aux arrondis. La méthode est donc heuristique. Cependant, la variante multimodulaire se prête à une implémentation similaire : l'idée est alors de coder les réduites modulo p par des matrices de flottants machine plutôt que d'entiers, et de faire à nouveau appel aux BLAS pour les multiplier [80].

FFT doubling. Dans le calcul d'arbres de produits, les techniques précédentes se combinent à une amélioration spécifique. Bernstein [24, §12.8] appelle cette idée *FFT doubling* et l'attribue à R. Kramer. Le point de départ est que la moitié des coefficients de la transformée de Fourier en taille $2\,n$ d'un tableau de n éléments (convenablement complété par des zéros) sont simplement ceux de la transformée en taille n du même tableau. Les n coefficients manquants se calculent eux-mêmes par une transformée en taille n. En un nœud interne d'un arbre de produits, la moitié des coefficients de chacune des deux transformées directes parmi les trois en lesquelles se décompose la multiplication sont donc déjà connues. La figure 7.3 illustre cela dans le cas des polynômes. Malheureusement, on ne sait apparemment pas faire mieux pour calculer les n coefficients manquants que de repasser par la représentation classique des polynômes ou des entiers. Le gain sur l'ensemble de l'algorithme de scindage binaire est donc seulement d'un facteur $3/2 + o(1)$.

Synthèse. En exploitant ces remarques sur le coût des multiplications aux nœuds, on arrive au raffinement suivant de la proposition 7.11.

PROPOSITION 7.15. Supposons $M(n) = 3\,F(2\,n) + O(n)$. Soit d' une borne sur les degrés de la matrice $B(n) \in \mathbb{A}[n]$ et du polynôme $q(n) \in \mathbb{Z}[n]$ de l'équation (7.3). Soit h' une borne sur leurs hauteurs. Lorsque $N = 2^K$ est une puissance de deux, le calcul de l'arbre de produits $P(0, N)$ sans mise en forme irréductible du résultat demande au plus

$$\left(\frac{2}{3} + o(1)\right) s^2\, M(N\,(h' + d' \lg N)) \lg N$$

opérations binaires quand $K, h', d', s \to \infty$ avec $s^{\omega-2} = o(K)$ et $\lg d' = O(K)$. Si en outre la fonction $n \mapsto M(n)/(n \lg n)$ est croissante et $\lg h' = o(\lg N)$, on peut remplacer la constante $2/3$ par $1/3$. ♢

DÉMONSTRATION. Notons $H = h' + d'\,K$. À profondeur $k < K$ dans l'arbre de produits (la profondeur de la racine étant zéro), on a à effectuer 2^k produits d'éléments de $\mathbb{A}^{s\times s} \times \mathbb{Z}$ de hauteur bornée par $H_k = 2^{K-k-1}\,H$. En utilisant la technique de *FFT*

6. *Basic Linear Algebra Subprograms*

doubling, chacun demande $4\,s^2+2$ transformées directes d'entiers de taille H_k, suivies de $O(H_k)$ produits ponctuels et de $2\,s^2+1$ transformées inverses en taille $2\,H_k$, soit un total de

$$(2\,s^2+1)\,(2\,F(H_k)+F(2\,H_k))+O\big((s^\omega+1)\,H_k\big)\leqslant\frac{4}{3}\,s^2M(H_k)\,(1+o(1))$$

opérations. Le coût total du scindage binaire est donc borné par

$$\left(\frac{4}{3}+o(1)\right)s^2\,M(H_0)\lg N\leqslant\left(\frac{2}{3}+o(1)\right)s^2\,M(N\,H)\lg N.$$

Sous l'hypothèse $\lg d'=o(K)$, il domine celui de l'évaluation des feuilles, estimé dans la preuve de la proposition 7.9 à $O(s^2\,2^K\,M(H_{K-1})\lg d')$.

Avec $M(n)=f(n)\,n\lg n$ où $f(n)$ est croissante, et $\lg h'=o(\lg N)$, on a

$$\begin{aligned}\sum_{k=0}^{K}2^k\,M(H_k)&=\sum_{k=0}^{K}f(H_k)\,H_0\,(\lg H_0-k+1)\\&\leqslant f(H_0)\,H_0\left(K\lg H+\frac{1}{2}\,K^2+O(K)\right)\\&\sim\frac{K}{2}\,M(H_0)\end{aligned}$$

(une remarque attribuée dans ce contexte à D. Stehlé par Zimmermann [267]), d'où le second résultat. □

L'hypothèse $\lg h'=o(\lg N)$ qui fait gagner un facteur $2+o(1)$ dans l'analyse est relativement contraignante. Parmi les applications à venir, elle s'applique à l'évaluation d'une fonction D-finie en un point rationnel fixé (§8.2), mais pas à l'algorithme *bit-burst* (§8.5). Concernant ce dernier, on pourrait tout de même prolonger l'analyse en explicitant la relation entre N et h'.

Discussion. Les techniques de réutilisation de transformées violent l'abstraction de la multiplication boîte noire. Cela les rend délicates à mettre en œuvre de façon robuste avec les interfaces offertes par les implémentations de la multiplication rapide. En outre, la plupart des utilisations du scindage binaire concernent des récurrences particulières et de petit ordre, pour lesquelles ces techniques n'apportent qu'une accélération modérée [262]. Pour ces différentes raisons, elles sont encore peu utilisées en pratique dans ce contexte. Cela pourrait évoluer : devant la multiplication des algorithmes rapides qui se placent dans le « modèle FFT » [24], on voit se dessiner dans le développement récent de bibliothèques comme GMP [116] ou FLINT [124] une volonté de les prendre en charge.

7.4.2 Multiplication boîte noire

Si l'on s'interdit de décomposer en opérations plus élémentaires la multiplication d'entiers, il reste l'opportunité d'appliquer des formules de multiplication rapide qui diminuent le nombre de multiplications de grands entiers à effectuer à chaque nœud, au prix d'additions et de multiplications par des constantes supplémentaires. Celles-ci ont une complexité linéaire et n'augmentent pas significativement la taille des opérandes auxquelles s'appliquent les multiplications « longues » restantes, d'où un gain sur la complexité binaire totale.

Algébriques et séries. L'exemple le plus évident concerne la multiplication dans $\mathbb{K}=\mathbb{Q}(i)$, qui ne prend que quatre multiplications entières par la formule « de Karatsuba » — apparemment due à Gauss dans ce cas —

$$\frac{x+i\,y}{w}\,\frac{x'+i\,y'}{w'}=\frac{(a-b)+i\,(c-a-b)}{w\,w'},\quad a=x\,x',\quad b=y\,y',\quad c=(x+y)\,(x'+y')$$

au lieu de cinq dans sa forme naïve. Plus généralement, on peut multiplier des éléments de hauteur bornée par h d'un corps de nombres de degré d en $2\,d\,M(h)+O(h)$ opération binaires en utilisant l'algorithme de Toom-Cook (contre $(d^2+1)\,M(h)$ naïvement). Il en va de même des séries tronquées qui interviennent dans les algorithmes des sections 8.3 à 8.6.

Matrices. La plupart des formules de multiplication rapide de matrices, comme celle classique de Strassen, sont, nous l'avons vu, des algorithmes bilinéaires (voir §7.2.4). Comme nous nous intéressons à des matrices sur un anneau commutatif, nous pouvons avoir recours à des algorithmes un peu moins contraints, dits *quadratiques* [44, §14.1], où les combinaisons linéaires qui fournissent les éléments à multiplier peuvent mêler les coefficients de A à ceux de B. Notamment, l'algorithme de Waksman [250], multiplie deux matrices $s\times s$ sur un anneau commutatif arbitraire en $s^2\lceil s/2\rceil+(2s-1)\lfloor s/2\rfloor$ multiplications scalaires.

Drevet *et al.* [77] recensent les bornes de complexité bilinéaire et quadratique pour le produit de matrices de petite taille que l'on peut obtenir en combinant judicieusement des formules connues. Le tableau 7.1 reproduit quelques-uns des résultats. Ils montrent en particulier que la combinaison des formules de Makarov [163], Waksman et Strassen peut être battue dès la taille 9. Cela réfute une observation énoncée dans mon article de présentation de NumGfun [174], sur la base d'un inventaire plus superficiel [173].

taille s	2	3	4	5	6	7	8	9	10	15	20	30
mul. $\mathrm{MM}(s)$	7	22	46	93	141	235	316	472	595	1941	4158	12710
$\mathrm{MM}(s)/s^3$	0,88	0,81	0,72	0,74	0,65	0,69	0,62	0,65	0,60	0,58	0,52	0,47

Tableau 7.1. Bornes sur le nombre de multiplications scalaires pour le produit de matrices carrées sur un anneau commutatif [77, Table 4].

Synthèse. On peut mener dans ce cas une analyse de complexité similaire à celle de la proposition 7.15. Reprenons les notations de sa preuve, mais puisque le modèle boîte noire est d'autant plus pertinent que s et d' sont petits, supposons-les cette fois fixés. Un produit dans $\mathbb{K}^{s\times s}\times\mathbb{Z}$ à profondeur k dans l'arbre prend maintenant au plus $(3\,\mathrm{MM}(s)+1)\,M(H_k)\,(1+o(1))$ opérations, où $\mathrm{MM}(s)$ représente la complexité multiplicative du produit de matrices de taille s et le facteur 3 provient de la formule de Karatsuba. Le coût total de l'algorithme est donc de

$$\left(\frac{3\,\mathrm{MM}(s)+1}{2\,C}+o(1)\right)M\big(N\,(h'+d'\lg N)\big)\lg N\qquad(N,h'\to\infty)$$

opérations binaires, où $C=1$ en général et $C=2$ dans les hypothèses d'application de la remarque de Stehlé.

Discussion. Attention cependant : l'étude précédente se place dans le cas de matrices *denses*. Sur le plan pratique, les formules de multiplication « rapide » peuvent être considérablement plus lentes que la méthode naïve quand les matrices à multiplier ont des coefficients nuls ou déséquilibrés.

Dans les différents algorithmes de ce chapitre, les matrices $B(n)$ dont on forme le produit sont des matrices compagnon ou possèdent un bloc compagnon de taille s sur la diagonale. Les produits partiels des s niveaux inférieurs de l'arbre de produits comportent donc un grand nombre de coefficients nuls. L'algorithme naïf de produit matriciel en tire mécaniquement parti : les multiplications de grands entiers par zéro sont essentiellement gratuites (et les multiplications par de « petites » constantes ont un coût linéaire, négligeable devant celui des multiplications entre grands entiers). Il faut en revanche se garder d'utiliser à ce stade des formules de multiplication rapide comme celle de Strassen qui ne tiennent pas compte des positions des zéros.

Autre exemple : multiplier aveuglément par la formule de Karatsuba des éléments de $\mathbb{A}$ qui s'avèrent en fait être réels fait perdre un facteur de l'ordre de 2 en temps par rapport à une multiplication réelle ou à une multiplication complexe naïve. Le même type de remarque s'applique aux matrices « particulières » de l'équation (7.7).

7.4.3 Complexité en espace

Une faiblesse relative du scindage binaire est sa complexité en espace $\Omega(n \lg n)$ pour un résultat souvent de taille $O(n)$ seulement. Plusieurs techniques ont été développées pour pallier cet inconvénient, essentiellement pour les récurrences du premier ordre. Mentionnons simplement les idées principales.

Reconstruction rationnelle. Supposons la suite u_n à coefficients rationnels. Si l'on sait *a priori* que u_n est de taille $O(n)$ plutôt que seulement $O(n \lg n)$, on peut calculer le numérateur et le dénominateur de u_n modulo un nombre premier de taille $\Theta(n)$, les multiplier et en déduire u_n par reconstruction rationnelle [48]. C'est le cas quand u_n est une somme partielle du développement en série d'une G-fonction de Siegel [25]. Le coût total des multiplications est $O(M(n) \lg^2 n)$, celui de la reconstruction rationnelle $O(M(n) \lg n)$ [251][7], mais la complexité en espace descend à $O(n)$.

Troncature. Calculer avec des approximations flottantes à précision $O(n)$ au lieu de rationnels dans les $O(\lg \lg n)$ niveaux supérieurs de l'arbre des produits est réputé donner de bons résultats. Ce fait est mentionné par plusieurs auteurs d'implémentations [115, 150] sans justification théorique des troncatures acceptables. Une prépublication très récente [261] propose une analyse détaille dans un cas particulier. Dans le cas de séries D-finies convergentes, cependant, la norme des matrices $B(n)$ des récurrences sur les coefficients est bornée quand $n \to \infty$, et celle d'un produit partiel $B(j-1) \cdots B(i)$ est donc $O(j-i)$. Il devrait être possible de rendre ces bornes effectives à partir des séries majorantes que nous utilisons déjà pour contrôler la précision du prolongement analytique, et ainsi de limiter la hauteurs des sous-produits à $O(n)$. Les détails restent à creuser.

Représentation factorisée. Dans certains cas particuliers, il s'avère profitable de représenter numérateurs et dénominateur des coefficients et sommes partielles d'une série à sommer sous forme factorisée, de façon à pouvoir en retirer rapidement les facteurs communs [49].

Je renvoie à l'article de Cheng *et al.* [49] pour un panorama plus complet.

7. Monagan [178] rapporte qu'un algorithme avec cette complexité était déjà implémenté dans Magma auparavant.

Chapitre 8

Prolongement analytique numérique à grande précision

« Quand on ne sait pas où l'on va, il faut y aller !!...
...et le plus vite possible. »

— Jacques Rouxel, *Les devises Shadok*

Nous en venons maintenant à l'évaluation numérique des fonctions D-finies proprement dite, et plus précisément à la question : une fonction D-finie étant spécifiée par des conditions initiales à l'origine, comment en déduire sa valeur en un point quelconque de son domaine de définition ? On envisage — dans la même idée de généralité — une évaluation à précision arbitraire, et le principal paramètre de complexité est la précision attendue pour le résultat. Une solution théorique à ce problème de complexité quasi-optimale est connue depuis le début des années 1990. J'en présente ici une version entièrement explicite, avec plusieurs améliorations et simplifications (superflues pour le résultat de complexité mais sensibles en pratique) ainsi qu'un grand nombre de détails utiles pour une implémentation complète qui sont omis dans les descriptions existantes. Comme le précédent, ce chapitre se trouve résumé dans l'article [174].

8.1 Introduction

Problématique. Comme nous l'avons relevé à plusieurs reprises, la manipulation de fonctions spéciales dans un logiciel de calcul formel, la combinatoire analytique ou d'autres applications de la D-finitude requièrent la capacité de calculer des valeurs de fonctions D-finies. L'optique de ce chapitre est de fournir des outils d'évaluation numérique à la fois utilisables en pratique et aussi généraux que possible, c'est-à-dire applicables

1. à toutes les fonctions D-finies ;
2. sur la totalité de leur domaine de définition ;
3. à précision arbitraire.

Ces outils sont en outre complètement automatiques, c'est-à-dire qu'ils se contentent en entrée de la spécification d'une fonction par une équation différentielle et des conditions initiales. Nul besoin, en particulier, « d'indications » pour régler la précision des calculs intermédiaires afin de garantir une certaine borne d'erreur au final.

Viser ainsi la généralité ne signifie en aucun cas que l'on cherche à couvrir toutes les *applications*, qui bien souvent demandent plutôt d'évaluer aussi efficacement que possible une petite classe de fonctions sur un domaine et à une précision fixées à l'avance. En revanche, disposer au préalable d'une implémentation générale est d'une grande aide pour en fabriquer, manuellement *ou automatiquement*, de plus spécialisées et plus efficaces [50, 155], ce qui justifie de commencer par là.

Évaluation de fonctions à grande précision. Des algorithmes appropriés existent dans la littérature. Il est connu notamment que la valeur prise par une fonction D-finie en un point quelconque de sa surface de Riemann peut être calculée à 2^{-n} près en $\tilde{O}(n)$ opérations binaires [57], toutes les bornes qui garantissent cette précision étant déterminées à la volée à partir de l'équation différentielle et des conditions

initiales [235][1]. Cette complexité est presque linéaire en la taille du résultat écrit en notation rationnelle ou en virgule fixe.

L'algorithme, dû à Chudnovsky et Chudnovsky, qui y aboutit se fonde sur la sommation de séries par scindage binaire présentée dans le chapitre précédent. Son coût est en général de $O(M(n \lg^2 n \lg\lg n))$ opérations binaires [235] (analyse améliorée à $O(M(n \lg^2 n))$ dans ce qui suit), et peut descendre à $O(M(n \lg n))$ pour des fonctions et des points particuliers, par exemple pour l'évaluation de la constante e.

Des algorithmes de complexité quasi-linéaire en la précision étaient connus auparavant, depuis une série de travaux de Brent [37, 38, 39] dans les années 1970, pour les fonctions élémentaires ainsi que diverses constantes classiques et fonctions spéciales (D-finies ou non). Certains sont à base de scindage binaire et représentent des cas particuliers de l'algorithme de Chudnovsky et Chudnovsky. D'autres ont une complexité asymptotique un peu meilleure. En particulier, l'itération de la moyenne arithmético-géométrique (AGM) permet de construire des algorithmes de complexité $O(M(n) \log n)$ pour un grand nombre de fonctions. Son application la plus fameuse est sans doute l'algorithme de Brent-Salamin [212, 39], qui fournit la meilleure complexité théorique[2] connue pour le calcul de π. Je renvoie aux livres de Borwein et Borwein [31] ainsi que de Brent et Zimmermann [40, chap. 4] pour un panorama.

Un certain nombre de ces algorithmes pour les fonctions élémentaires sont utilisés dans des bibliothèques comme MPFR [183, 96], et à travers elles par les systèmes de calcul formel. Au contraire, comme le relève Dupont [81, §9.2.1], l'algorithme général pour les fonctions D-finies est demeuré théorique. Malgré les capacités bien réelles (cf. §2.2) des outils de calcul formel concernant ces fonctions, il n'en existait pas d'implémentation avant la présente étude. Tandis que les mêmes logiciels fournissent des routines d'évaluation numérique à précision arbitraire, écrites une par une, pour des dizaines de fonctions élémentaires et fonctions spéciales, l'algorithme de Chudnovsky et Chudnovsky ou ses variantes n'avaient servi au mieux que de « recettes » à spécialiser pour obtenir un algorithme applicable à une fonction donnée.

Contenu du chapitre. Ce chapitre présente le « gros morceau » de NumGfun, à savoir le prolongement analytique numérique à grande précision, qui vise à combler ce manque. À ce titre, il contient :

- des rappels complets sur les algorithmes mis en œuvre ;
- plusieurs petites améliorations à ceux-ci ou à leur analyse (la technique de calcul simultané des dérivées de la section 8.4, celle de calcul rapide des composantes des solutions généralisées de la section 8.6.2, ainsi que la proposition 8.18 sont apparemment nouvelles) ;
- et une description assez détaillée de l'implémentation, accompagnée d'exemples, qui couvre notamment la propagation des erreurs au cours des calculs intermédiaires.

L'analyse de la complexité des algorithmes vis-à-vis des paramètres autres que la précision est aussi poussée un peu plus loin que dans les travaux antérieurs.

La méthode de scindage binaire vue au chapitre précédent induit immédiatement un algorithme pour évaluer les séries D-finies à l'intérieur de leur disque de convergence, auquel nous consacrons une courte section 8.2. Nous revenons ensuite en détail sur l'application de la même idée pour le *prolongement analytique* numérique, qui offre un moyen d'évaluer une série D-finie en-dehors du disque de convergence de son développement en série entière (sections 8.3 et 8.4). Le prolongement analytique aboutit aussi à l'algorithme *bit burst*, qui fournit la borne de complexité quasi-linéaire pour l'évaluation des fonctions D-finies dans le cas général (section 8.5). La section 8.6

1. Nous avons déjà vu aux chapitres 5 et 6 un raffinement de l'algorithme de calcul de bornes sous-jacent.

2. Les records de calcul récents [17, 262] utilisent la formule (6.1) couplée au scindage binaire, d'un coût plus élevé d'un facteur logarithmique, mais avec une « constante » beaucoup plus faible en pratique. L'avantage pratique vient notamment de ce que seule une petite partie du calcul par scindage binaire a lieu à la pleine précision du résultat.

décrit la généralisation de cette approche pour effectuer la connexion entre points ordinaires et singuliers réguliers, dans une nouvelle variante qui exploite le formalisme du chapitre 4. Enfin, je présente plusieurs applications en section 8.7. Il s'agit d'exemples issus de classes de problèmes dont le traitement pourrait être automatisé, mais n'est pas ou pas complètement implémenté dans NumGfun pour l'instant.

Convention 8.1. On reprend ici toutes les conventions énoncées en §7.2.1, au début du chapitre précédent. ♢

8.2 Évaluation d'une série D-finie dans son disque de convergence

La suite des coefficients de Taylor d'une fonction D-finie étant P-récursive, la sommation de séries par scindage binaire fournit un algorithme efficace pour évaluer ces fonctions à grande précision dans le disque de convergence de leurs développements en série entière. « Grande précision » signifie essentiellement que nous allons mesurer la complexité de l'algorithme en fonction de la borne d'erreur 10^{-p} imposée au résultat, en faisant tendre p vers l'infini.

L'idée, apparemment due à Schroeppel, semble remonter au *Hakmem* [15, §178], qui ne fait cependant qu'énoncer une complexité sans algorithme[3]. À nouveau, sa formulation générale se trouve dans les travaux des frères Chudnovsky. Une variante a été développée indépendamment par E. Karatsuba sous le nom de *fast E-function evaluation* ou FEE [138], avec plusieurs applications intéressantes.

Proposition 8.2. [57, Theorem 3.2] Soient y une fonction D-finie sur $\mathbb{K}$ et $\rho \in]0, +\infty]$ le rayon de convergence du développement de Taylor de y à l'origine. Fixons $\mu < \rho$. Pour $z \in \mathbb{K}$ tel que $0 < |z| \leqslant \mu$ et de hauteur bornée par h, on peut calculer une approximation de $y(z)$ à précision 10^{-p} en

$$\begin{cases} O\Big(M\Big(\dfrac{p\,(h+\lg p)}{\lg\,(\rho/|z|)}\Big)\lg\,(p\,h)\Big) & \text{si } \rho<\infty \\ O\Big(M\big(p\,(h+\lg p)\big)\lg h\Big) & \text{si } \rho=\infty \end{cases} \tag{8.1}$$

opérations binaires quand $h, p \to \infty$. (À h et μ fixés, la borne est uniforme en z.) ♢

Démonstration. La suite $(y_n\, z^n)_{n\in\mathbb{N}}$ satisfait une relation de récurrence linéaire de hauteur $O(h)$. D'après la proposition 7.11, $y(\zeta)$ se calcule donc en

$$O\Big(M\big(N\,(h+\lg N)\big)\lg\,(N\,h)\Big)$$

opérations binaires, où N est choisi de sorte que $|y_{N;}(\zeta)| \leqslant 10^{-p}$. Le tableau 6.1 donne $N \sim \dfrac{p}{\lg\,(\rho/|z|)}$ si $\rho<\infty$ et $N \sim \dfrac{p}{\tau \lg p}$ pour un certain τ (égal à $-\kappa$ dans les notations du chapitre 5) sinon. □

Nous écrirons explicitement l'algorithme dans un cadre plus général un peu plus loin. Concrètement, les valeurs de p ciblées par l'implémentation dans NumGfun vont de quelques centaines à quelques millions.

3. Gosper [112, p. 263] écrit à propos de l'idée de sommer des séries en formant un produit de matrices par scindage binaire :

> Now suppose that you want a very precise value of $\zeta(3)$, and thus wish to sum n terms of this series. You will find it dramatically cheaper to write out instead the first n matrices, and then pairwise multiply to form $\lceil n/2 \rceil$ products, and repeat until only one matrix remains. (In 1985, I used this technique, essentially due to R. Schroeppel, to temporarily steal the π computation record from Japan.)

Cela tend à confirmer que c'est bien là une des techniques qu'avaient en tête les auteurs de l'item 178 du *HAKMEM*.

EXEMPLE 8.3. Calculons la valeur au point $x=1$ de la fonction d'erreur

$$\operatorname{erf} x = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\,\mathrm{d}t = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n\, x^{2n+1}}{(2\,n+1)\,n!} \tag{8.2}$$

à 10^{-10^6} près :

```
> deq := holexprtodiffeq(erf(x), y(x));
```

$$\left\{2\left(\frac{\mathrm{d}}{\mathrm{d}x}y\left(x\right)\right)x+\frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{\mathrm{d}}{\mathrm{d}x}y\left(x\right)\right),\, y\left(0\right)=0,\left(y'\right)\left(0\right)=\frac{2}{\sqrt{\pi}}\right\}$$

```
> infolevel[gfun] := 2:
> evaldiffeq(deq, y(x), 1, 1000000);
   ordinary_step_transition_matrix:
   "0 --> 1 (1 derivative[s]), prec~=.88623e-1000001, #terms=434626"
   0.84270079294971486934122063508...12221396220957362845496953847529 1
```

Le message d'information indique que le calcul a été réalisé directement, en sommant les termes jusqu'à x^{434626} de la série (8.2) — une troncature choisie sur la base des bornes établies aux chapitres précédents. Le calcul prend 133,8 secondes.

En comparaison, la commande Maple (qui utilise du code spécialisé pour la fonction erf, sans optimisation pour les arguments rationnels)

```
> evalf[1000000](erf(1));
```

donne le même résultat après 262,1 secondes. Avec Sage 4.7 [225], qui s'appuie pour cela sur la bibliothèque MPFR [183] (et indirectement sur MPIR [184], elle-même dérivée d'une version de GMP plus récente que celle utilisée par Maple), le calcul à une précision comparable de $3{,}32\cdot 10^6$ bits

```
sage: erf(RealField(3320000)(1))
```

dure 924 secondes. ♢

Des précisions de cet ordre sont utiles pour certaines applications en théorie des nombres [57] ou encore pour construire des tests d'égalité heuristiques [236, §5]. Il peut aussi arriver que la façon la plus simple d'obtenir au final un résultat à précision modérée passe par des calculs intermédiaires à grande précision, particulièrement si la correction du résultat dépend de bornes d'erreur pessimistes. Bailey *et al.* [10] présentent une sélection d'applications du calcul numérique à grande précision en physique et en mathématiques expérimentales.

Soulignons en passant que les estimations de complexité de la proposition 8.2, comme celles des autres algorithmes rapides mentionnés en introduction, sont données en fonction du coût de la multiplication d'entiers. Le caractère quasi-linéaire de tous ces algorithmes, que ce soit en théorie ou en pratique, dépend de façon cruciale de la multiplication rapide.

8.3 Prolongement analytique

8.3.1 Principe

Le formalisme que nous adoptons est directement inspiré de celui de van der Hoeven [236, 239]. Fixons une équation différentielle linéaire homogène

$$L(z,\partial)\cdot y(z) = a_r(z)\,y^{(r)} + \cdots + a_1(z)\,y'(z) + a_0(z)\,y(z) = 0, \quad a_k \in \mathbb{K}[z], \tag{8.3}$$

avec $a_r(z)\neq 0$. On note aussi $D = L(z,\partial)$. Soit $S = \operatorname{Sing}(D) \subset \bar{\mathbb{Q}}$ l'ensemble (fini) des points singuliers de (8.3).

Comme énoncé en §3.2.2, les solutions de (8.3) définies au voisinage de l'origine s'étendent par prolongement analytique au revêtement universel de $\mathbb{C}\setminus S$. On peut les évaluer en bonne complexité en des points quelconques par une version numérique

de ce processus de prolongement analytique, bâtie sur l'algorithme d'évaluation dans le disque de convergence présenté à la section précédente [54, 57, 56]. L'idée est simplement de transporter, en plusieurs pas, les conditions initiales du point où elles sont spécifiées au point d'évaluation, chaque pas consistant à sommer des séries solutions à l'intérieur de leur disque de convergence (voir figure 8.1).

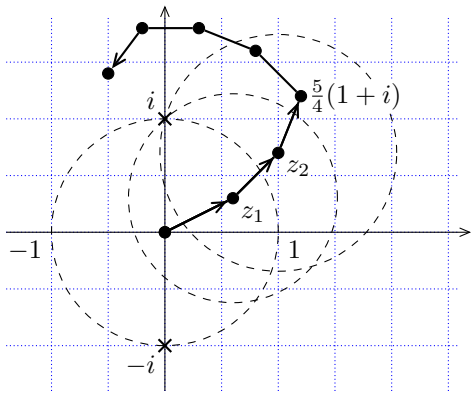


Figure 8.1. Exemple de prolongement analytique.

Le recours au prolongement analytique numérique est indépendant du fait de calculer les sommes de séries en question par scindage binaire, ou même de se concentrer sur les équations différentielles linéaires à coefficients polynomiaux. Il s'apparente à des méthodes classiques d'intégration numérique d'équations différentielles [14, 71, 189], souvent utilisées conjointement avec l'arithmétique d'intervalles afin d'encadrer rigoureusement les solutions. D'ordinaire, la précision de représentation des coefficients numériques et l'ordre des développements en série sont fixés. Le découpage du chemin d'intégration en plusieurs pas vise autant sinon plus à obtenir des encadrements fins qu'à sortir du disque de convergence des développements en série initiaux.

Solutions canoniques. Concrètement, on réécrit l'équation homogène (8.3) sous forme matricielle

$$Y'(z) = A(z)\,Y(z), \qquad Y(z) = \begin{pmatrix} y(z) \\ y'(z) \\ \vdots \\ \frac{1}{(r-1)!}\,y^{(r-1)}(z) \end{pmatrix}. \tag{8.4}$$

Le choix du vecteur $Y(z)$ induit celui, pour tout $z_0 \in \mathbb{C} \setminus S$, d'une famille de solutions canoniques de (8.3) définies par les conditions initiales en z_0

$$y_{[z_0,j]}(z) = (z-z_0)^j + O\big((z-z_0)^r\big), \qquad 0 \leqslant j \leqslant r-1. \tag{8.5}$$

Elles forment une base des solutions de (8.3) au voisinage de z_0, qui n'est autre que celle déjà choisie comme base canonique en fin de section 3.2.2. Le vecteur

$$\boldsymbol{y}_{[z_0]}(z) = \begin{pmatrix} y_{[z_0,0]}(z) & y_{[z_0,1]}(z) & \cdots & y_{[z_0,r-1]}(z) \end{pmatrix} \tag{8.6}$$

des solutions canoniques en z_0 est la première ligne de la matrice fondamentale $\boldsymbol{Y}_{[z_0]}(z)$ de (8.4) telle que $\boldsymbol{Y}_{[z_0]}(z_0) = \mathrm{Id}$. Nous placerons les paramètres $[z_0, i]$ indifféremment en indice ou sur la ligne de base.

Chemins et matrices de transition. Si γ est un arc tracé dans $\mathbb{C} \setminus S$, on écrit $z_0 \xrightarrow{\gamma} z_1$ pour rappeler l'origine z_0 et la destination z_1 de γ. On note simplement $z_0 \to z_1$ le chemin en ligne droite de z_0 à z_1. La concaténation de deux chemins $z_0 \xrightarrow{\gamma} z_1$ et $z_1 \xrightarrow{\gamma'} z_2$ est notée $z_0 \xrightarrow{\gamma} z_1 \xrightarrow{\gamma'} z_2$.

Par linéarité, pour tout arc (polygonal) $z_0 \overset{\gamma}{\to} z_1$ tracé dans $\mathbb{C} \setminus S$, il existe une *matrice de transition*, ou *matrice résolvante* $M(z_0 \overset{\gamma}{\to} z_1)$ qui « transporte les conditions initiales » (et, dualement, les solutions) le long du chemin suivant les relations

$$Y(z_1) = M(z_0 \overset{\gamma}{\to} z_1)\, Y(z_0), \qquad \boldsymbol{y}_{[z_0]}(z) = \boldsymbol{y}_{[z_1]}(z)\, M(z_0 \overset{\gamma}{\to} z_1), \tag{8.7}$$

où les évaluations en z_1 font référence à la valeur de Y obtenue par prolongement analytique le long du chemin $z_0 \overset{\gamma}{\to} z_1$. Explicitement, elle s'écrit

$$M(z_0 \overset{\gamma}{\to} z_1) = \boldsymbol{Y}_{[z_0]}(z_1) = \begin{pmatrix} y_{[z_0,0]}(z_1) & \cdots & y_{[z_0,r-1]}(z_1) \\ \vdots & & \vdots \\ \frac{1}{(r-1)!}\, y^{(r-1)}_{[z_0,0]}(z_1) & \cdots & \frac{1}{(r-1)!}\, y^{(r-1)}_{[z_0,r-1]}(z_1) \end{pmatrix}, \tag{8.8}$$

avec le même abus de notation. Les matrices de transition se composent naturellement : on a

$$M(z_0 \overset{\gamma}{\to} z_1 \overset{\gamma'}{\to} z_2) = M(z_1 \overset{\gamma'}{\to} z_2)\, M(z_0 \overset{\gamma}{\to} z_1), \qquad M(z_1 \overset{\gamma}{\to} z_0) = M(z_0 \overset{\gamma}{\to} z_1)^{-1} \tag{8.9}$$

pour tous chemins $z_0 \overset{\gamma}{\to} z_1$ et $z_1 \overset{\gamma'}{\to} z_2$. S'il est besoin de préciser l'opérateur auquel est associée une matrice de transition $M(z_0 \overset{\gamma}{\to} z_1)$, nous écrirons aussi $M(D, z_0 \overset{\gamma}{\to} z_1)$.

Esquisse de l'algorithme. L'évaluation du prolongement analytique le long d'un chemin $z_0 \overset{\gamma}{\to} z$ donné d'une fonction D-finie procède comme suit :

1. on remplace le chemin par une ligne brisée $z_0 \to z_1 \to \cdots \to z_m$ où $z_m = z$ qui lui est homotope dans $\mathbb{C} \setminus S$, telle que $z_{\ell+1}$ soit à l'intérieur du disque de convergence du développement en série en z_ℓ de $\boldsymbol{Y}_{[z_\ell]}$ pour tout ℓ ;
2. pour chaque ℓ, on calcule une approximation $\tilde{M}_\ell \in \mathbb{K}^{r\times r}$ de la matrice de transition $M_\ell = M(z_\ell \to z_{\ell+1})$;
3. on forme le produit $\tilde{M}_{m-1}\, \tilde{M}_{m-2} \cdots \tilde{M}_0\, \tilde{U}$, où $\tilde{U} \in \mathbb{K}^r$ est une approximation du vecteur de conditions initiales $Y(z_0)$.

EXEMPLE 8.4. Rappelons que la fonction arctangente est définie sur le plan complexe fendu le long des demi-droites $[i, i\,\infty)$ et $[-i, -i\,\infty)$ et présente des singularités en $\pm i$. Elle satisfait l'équation différentielle homogène $(1+z^2)\, y''(z) + 2\, z\, y'(z) = 0$, avec les conditions initiales $y(0) = 0$ et $y'(0) = 1$. La base de solutions canoniques à l'origine de cette équation différentielle est $(1, \arctan)$.

Soit à calculer $\arctan z$ pour $z = \frac{5}{4}(1+i)$. Le disque de convergence du développement de Taylor à l'origine de arctan étant limité par les singularités en $\pm i$, on ne peut se contenter de sommer ce développement. Choisissons deux points intermédiaires $z_1 = \frac{3}{5} + \frac{3}{10}\, i$, $z_2 = 1 + \frac{7}{10}\, i$ (voir figure 8.1). On a alors $|z_1| < 1$, $|z_2 - z_1| < |i - z_1|$ et $|z_3 - z_2| < |i - z_2|$, autrement dit, chacun des points z_1, z_2, z est situé à l'intérieur du disque de convergence des séries solutions de l'équation en son prédécesseur le long du chemin. On calcule les matrices de transition

$$M_{0\to z_1} = \begin{pmatrix} 1 & \arctan z_1 \\ 0 & (1+z_1^2)^{-1} \end{pmatrix} \approx \begin{pmatrix} 1 & 0{,}57052 + 0{,}22009\, i \\ 0 & 0{,}72884 - 0{,}20660\, i \end{pmatrix}$$

$$M_{z_1\to z_2} \approx \begin{pmatrix} 1 & 0{,}39020 + 0{,}24502\, i \\ 0 & 0{,}57114 - 0{,}29113\, i \end{pmatrix} \quad M_{z_2\to z} \approx \begin{pmatrix} 1 & 0{,}30390 + 0{,}37521\, i \\ 0 & 0{,}54665 - 0{,}30827\, i \end{pmatrix},$$

dont chacune des entrées est la somme d'une série numérique convergente. L'entrée en haut à droite du produit $M_3\, M_2\, M_1$ donne la valeur de $\arctan z$.

On peut étendre le prolongement analytique numérique au-delà de la ligne de coupure $[i, i\,\infty)$. Ce que l'on calcule alors est une autre détermination de la « fonction multivaluée » arctangente, en l'occurrence (dans l'exemple du dessin), $\pi + \arctan$. ♢

Le calcul de la matrice de transition le long de chaque pas $z_\ell \to z_{\ell+1}$ relève d'une variante de l'algorithme d'évaluation esquissé en §8.2. Mais avant d'y revenir en §8.4, voyons comment on combine les résultats afin d'obtenir la matrice de transition le long d'un chemin quelconque.

8.3.2 Algorithme de suivi d'un chemin

Les algorithmes 8.5 et 8.6 calculent la matrice de transition le long d'un chemin comme indiqué ci-dessus. Toute la suite de cette section est consacrée à détailler différents aspects de leur fonctionnement.

Conditions initiales et produit des matrices de transition. L'algorithme 8.6 prend en entrée une matrice non nécessairement carrée de conditions initiales U ainsi qu'un chemin $z_0 \overset{\gamma}{\to} z_{\text{fin}}$, et calcule une approximation de $M(z_0 \overset{\gamma}{\to} z_{\text{fin}})\,U$. Pour calculer la matrice de transition $M(z_0 \overset{\gamma}{\to} z_{\text{fin}})$, on adopte $U = \mathrm{Id}$; pour évaluer une solution donnée, $U = Y(0)$. Si l'on ne s'intéresse qu'à la valeur de y en z_{fin} et non celles de ses dérivées, on peut ne calculer que la première ligne de $\tilde{M}_{m-1}$, ce qui représente un gain sensible lorsque le chemin est réduit à un pas. Le produit $\tilde{M}_{m-1} \cdots \tilde{M}_0$ est lui-même calculé par scindage binaire.

Choix des points intermédiaires. Il est nécessaire de découper le chemin donné en entrée en pas assez courts pour que chacune des matrices $\tilde{M}_\ell$ se calcule en sommant des séries convergentes. Le coût de chaque pas $z_\ell \to z_{\ell+1}$ est lié au rapport

$$|z_{\ell+1} - z_\ell| / \mathrm{dist}(z_\ell, S)$$

(proposition 8.2), de sorte que l'on a intérêt à choisir la longueur des pas de manière à minimiser le coût total. Le prolongement analytique s'avère profitable même pour évaluer une série *à l'intérieur* de son disque de convergence.

Exemple 8.7. Nous avons observé au chapitre 1 que NumGfun permettait d'évaluer au point $z = -0{,}99$ une fonction de Heun doublement confluente, singulière en $z = -1$, en un temps acceptable. Sur la base des bornes des chapitres précédents, un calcul direct à l'aide de la série de Taylor à l'origine pour obtenir le résultat à précision 10^{-400} aurait conduit à sommer plus de 225000 termes de la série. La procédure donnée par `diffeqtoproc` effectue en fait automatiquement un prolongement analytique le long du chemin

$$0 \to -\frac{1}{2} \to -\frac{3}{4} \to -\frac{22}{25} \to -\frac{47}{50} \to -\frac{97}{100} \to -\frac{99}{100},$$

ce qui réduit le nombre de termes à chaque pas à environ 1700 (un peu plus pour le dernier, où l'on ne calcule qu'une ligne de la matrice de transition). ♢

Mieux, on peut chercher à déformer le chemin par homotopie de manière à minimiser le coût total, et partager le cas échéant les portions de chemin parcourues plusieurs fois. On ne dispose que de solutions partielles à ce problème de prolongement analytique optimal, qui couvrent toutefois les principaux cas intéressants. Le lecteur pourra consulter [54, §4], [241, §5.3], [173, §4.3] pour plus de détails. L'implémentation dans NumGfun se limite pour l'instant à une heuristique simple fondée sur ces idées.

8.3.3 Contrôle de l'erreur globale

Arrondi final. À partir de la précision cible 10^{-p} imposée par l'utilisateur, on commence par régler la précision des calculs intermédiaires afin de pouvoir arrondir leur résultat à un nombre décimal sans introduire d'erreur. Il s'agit de trouver ε tel que l'arrondi $\diamond(r) \in 10^{-p}\,\mathbb{A}$ d'une approximation $r \in \mathbb{K}$ à ε près du résultat exact $y(z)$ satisfasse $|\diamond(r) - y(z)| \leqslant 10^{-p}$.

ALGORITHME 8.5. PathTransitionMatrix$(D, [z_0, ..., z_m], \varepsilon)$

Entrée. Un opérateur différentiel $D \in \mathbb{Q}[z]\langle\partial\rangle$ d'ordre r. Une liste $[z_0, ..., z_m]$ de points de $\mathbb{K}$. Une précision ε.

Sortie. Une approximation $\tilde{M} \in \mathbb{K}^{r\times r}$ de la matrice $M = M(D, z_1 \to \cdots \to z_m)$ avec $\|\tilde{M} - M\|_{\mathrm{F}} \leqslant \varepsilon$.

Notes. Pour chaque $\ell \in [\![0, m-1]\!]$, on suppose précalculée une série majorante g_ℓ telle que $y[z_\ell, j] \trianglelefteqslant g_\ell$ pour $0 \leqslant j < r$. Toutes les opérations sur des éléments de $\mathbb{K}$ s'effectuent sans simplification des fractions.

1 si $m = 0$
  2 renvoyer la matrice identité
3 sinon, si $m = 1$
  4 renvoyer StepTransitionMatrix$(D, z_0, z_1, \varepsilon)$ (algorithme 8.12)
5 sinon
  6 calculer $B_2 \geqslant \prod_{\ell=\lfloor m/2\rfloor}^{m-1} \gamma_\ell$, où γ_ℓ est défini par (8.11)
  7 $\tilde{M}_1 :=$ PathTransitionMatrix$(D, [z_0, ..., z_{\lfloor m/2\rfloor}], \varepsilon/2B_2)$
  8 calculer $B_1 \geqslant \|\tilde{M}_1\|_{\mathrm{F}}$
  9 $\tilde{M}_2 :=$ PathTransitionMatrix$(D, [z_{\lfloor m/2\rfloor}, ..., z_m], \varepsilon/2B_1)$
  10 renvoyer la fraction non simplifiée $\tilde{M}_2\,\tilde{M}_1$ ♢

ALGORITHME 8.6. AnalyticContinuation$\big(D, z_0 \xrightarrow{\gamma} z_{\mathrm{fin}}, U, p\big)$

Entrée. Un opérateur $D \in \mathbb{Q}[z]\langle\partial\rangle$ d'ordre r. Un chemin $z_0 \xrightarrow{\gamma} z_{\mathrm{fin}}$ tracé dans $\mathbb{C} \setminus \mathrm{Sing}(D)$. Une matrice $U \in \mathbb{C}^{r\times t}$ de conditions initiales calculable à précision arbitraire. Un entier $p \in \mathbb{N}$.

Sortie. Une approximation $\tilde{\boldsymbol{Y}} \in 10^{-p}\,\mathbb{A}$ de la la matrice fondamentale $\boldsymbol{Y}(z)$ de D définie par $\boldsymbol{Y}(z_0) = U$, prise à l'extrémité z_{fin} du chemin (cf. prop. 8.11).

Note. Pour chaque $\ell \in [\![0, m-1]\!]$, on suppose précalculée une série majorante g_ℓ telle que $y[z_\ell, j] \trianglelefteqslant g_\ell$ pour $0 \leqslant j < r$.

1 subdiviser $z_0 \xrightarrow{\gamma} z_{\mathrm{fin}}$ en un chemin $z_0 \to \cdots \to z_m$ tel que $|z_{\ell+1} - z_\ell| < \mathrm{dist}(z_\ell, S)$ pour tout $\ell \in [\![0, m-1]\!]$, avec $z_m = z_{\mathrm{fin}}$
2 $\varepsilon := 10^{-(p+1)}$
3 calculer $B_{\mathrm{prol}} \geqslant \prod_{\ell=0}^{m-1} \gamma_\ell$, où γ_ℓ est défini par (8.11)
4 calculer $\tilde{U} \in \mathbb{K}^{r\times t}$ telle que $\|\tilde{U} - U\|_{\mathrm{F}} \leqslant \varepsilon/2B_{\mathrm{prol}}$
5 calculer $B_{\mathrm{ini}} \geqslant \|\tilde{U}\|_{\mathrm{F}}$
6 $\tilde{M} :=$ PathTransitionMatrix$(D, [z_0, ..., z_m], \varepsilon/2B_{\mathrm{ini}})$
7 $P := \tilde{M}\tilde{U}$ (calculé sans réduction au même dénominateur)
8 renvoyer l'approximation $\tilde{\boldsymbol{Y}} = P^*$ donnée par le lemme 8.8 appliqué coefficient par coefficient ♢

LEMME 8.8. Soit $p \in \mathbb{N}$. Pour tout $x = a/b \in \mathbb{Q}$, on note

$$x^* = \mathrm{sgn}(x) \diamond\left(10^{-(p+1)} \left\lfloor \frac{|a|\,10^{p+1}}{|b|} \right\rfloor, 10^{-p}\right),$$

où $\diamond(x, \varepsilon)$ représente l'arrondi au multiple de ε le plus proche. Si $z = x + i\,y \in \mathbb{K}$, le nombre $z^* = x^* + i\,y^* \in 10^{-p}\,\mathbb{A}$ satisfait $|z - z^*| \leqslant 0{,}85 \cdot 10^{-p}$. Pour $|z|$ borné, z^* se calcule en $O(M(p))$ opérations. ♢

DÉMONSTRATION. Quel que soit x, on a $|10^{p+1}\,|x| - \lfloor 10^{p+1}\,|a|/|b|\rfloor| < 1$, d'où

$$\big||x| - |x^*|\big| \leqslant 10^{-(p+1)} + \frac{1}{2}\,10^{-p} = \frac{6}{10}\,10^{-p},$$

et $|x - x^*| \leqslant 6/10 \cdot 10^{-p}$ puisque x et x^* ont même signe. Cela entraîne

$$|z - z^*| \leqslant \sqrt{2}\,\frac{6}{10}\,10^{-p} < 0{,}85 \cdot 10^{-p}.$$

Le calcul ne demande que des multiplications et divisions d'entiers de taille $O(p)$, et notamment, pas de pgcd. □

C'est la seule erreur *d'arrondi* à proprement parler, le reste du calcul étant effectué en arithmétique exacte. Il nous faut en revanche fixer la précision du calcul de chacune des matrices de transition intermédiaires de façon à garantir la borne $|r - y(z)| \leqslant \varepsilon$.

Norme de Frobenius. Comme il est d'usage dans ce type d'application, on norme les espaces de matrices par la norme de Frobenius, dont l'estimation est aisée et numériquement stable.

Définition 8.9. On appelle *norme de Frobenius* d'une matrice (non nécessairement carrée) $A = (a_{i,j})$ la quantité

$$\|A\|_{\mathrm{F}} = \sqrt{\sum_{i,j} |a_{i,j}|^2}. \qquad \diamondsuit$$

Les propriétés suivantes sont immédiates.

Proposition 8.10. La norme de Frobenius définit une norme sur $\mathbb{C}^{p\times q}$ pour tous p et q. Si $A \in \mathbb{C}^{p\times q}$ et $B \in \mathbb{C}^{p\times r}$ sont deux matrices de dimensions compatibles, on a

$$\|A\,B\|_{\mathrm{F}} \leqslant \|A\|_{\mathrm{F}}\,\|B\|_{\mathrm{F}}.$$

En particulier, la norme de Frobenius sur les matrices carrées est une norme d'algèbre. Lorsque $A \in \mathbb{C}^{r\times r}$, on a de plus

$$\|A\|_\infty \leqslant |||A|||_2 \leqslant \|A\|_{\mathrm{F}} \leqslant r\,\|A\|_\infty,$$

où $\|\cdot\|_\infty$ est la norme uniforme coefficient par coefficient et $|||\cdot|||_2$ la norme matricielle subordonnée à la norme hermitienne sur $\mathbb{C}^r$. Si F et G sont deux matrices de séries telles que $F \trianglelefteqslant G$, on a $\|F(z)\|_{\mathrm{F}} \leqslant \|G(|z|)\|_{\mathrm{F}}$ en tout point z du disque de convergence de G. □

Pour majorer la norme de Frobenius d'une matrice à coefficients dans $\mathbb{K}$, il suffit de remplacer chaque coefficient par son approximation flottante

$$\frac{\pm x \pm i\,y}{d} \approx \triangle\left(\frac{\pm\triangle(x) \pm i\,\triangle(y)}{\triangledown(d)}\right)$$

où $\triangle$ et $\triangledown$ indiquent respectivement un arrondi par excès et par défaut. On calcule ensuite la norme de la matrice obtenue par la formule de la définition 8.9, en arrondi vers $+\infty$.

Erreur cumulée. Fixons ℓ, et soit $g_\ell \in \mathbb{R}_+[[z]]$ une série majorante commune des solutions canoniques $y[z_\ell, i]$. On a alors

$$M(z_\ell \to z) = \boldsymbol{Y}[z_\ell](z) \trianglelefteqslant G_\ell(z) = \left(\ \tfrac{1}{i!}\, g_\ell^{(i)}(z)\ \right)_{0\leqslant i,j<r}, \tag{8.10}$$

d'où

$$\|M_\ell\|_{\mathrm{F}} \leqslant \|G_\ell(|z_{\ell+1} - z_\ell|)\|_{\mathrm{F}} = \left(r \sum_{i=0}^{r-1} \left(\tfrac{1}{i!}\, g_\ell^{(i)}(|z_{\ell+1} - z_\ell|)\right)^2\right)^{\frac{1}{2}}. \tag{8.11}$$

Lorsqu'on dispose déjà d'une valeur approchée $\tilde{M}_\ell$ de M_ℓ avec $\|M_\ell - \tilde{M}_\ell\|_{\mathrm{F}} \leqslant \varepsilon$ au moment où une borne sur sa norme est requise, on peut bien sûr remplacer (8.11) par $\|M_\ell\|_{\mathrm{F}} \leqslant \|\tilde{M}_\ell\|_{\mathrm{F}} + \varepsilon$. Si $\lg \|G_\ell(|z_{\ell+1} - z_\ell|)\|_{\mathrm{F}}$ est non-négligeable devant la précision du calcul, il peut éventuellement être rentable de faire tout le calcul à petite précision avec les bornes (8.11), puis d'utiliser les normes des matrices $\tilde{M}_\ell$ ainsi obtenues dans le calcul à grande précision.

PROPOSITION 8.11. La matrice $\tilde{\boldsymbol{Y}} = \text{AnalyticContinuation}(D,\, z_0 \xrightarrow{\gamma} z_1,\, I,\, p)$ calculée par l'algorithme 8.6 satisfait $\big\|\tilde{\boldsymbol{Y}} - \boldsymbol{Y}(z_1)\big\|_\infty \leqslant 10^{-p}$, où $\|\cdot\|_\infty$ représente la norme uniforme coefficient par coefficient. ♢

La borne est coefficient par coefficient et non en norme de Frobenius. Il n'est pas possible en effet d'approcher une matrice quelconque par une matrice de dénominateur 10^p, en norme de Frobenius, avec une erreur elle-même bornée par 10^{-p}.

DÉMONSTRATION. On vérifie tout d'abord par récurrence sur m que l'algorithme 8.5 fonctionne comme spécifié. C'est clair si $m=0$ ou $m=2$. Pour $m \geqslant 2$, posons

$$M_1 = M(D, z_0 \to \cdots \to z_{\lfloor m/2 \rfloor}), \qquad M_2 = M(D, z_{\lfloor m/2 \rfloor} \to \cdots \to z_m).$$

On a $\|M_2\|_{\mathrm{F}} \leqslant B_2$ d'après (8.9) et (8.11), et $\|\tilde{M}_1\|_{\mathrm{F}} \leqslant B_1$, donc

$$\big\|\tilde{M} - M\big\|_{\mathrm{F}} = \big\|\tilde{M}_2\,\tilde{M}_1 - M_2\,M_1\big\|_{\mathrm{F}} \leqslant \big\|\tilde{M}_2 - M_2\big\|_{\mathrm{F}}\big\|\tilde{M}_1\big\|_{\mathrm{F}} + \|M_2\|_{\mathrm{F}}\big\|\tilde{M}_1 - M_1\big\|_{\mathrm{F}} \leqslant \varepsilon.$$

Concernant l'algorithme 8.6, le même raisonnement justifie que

$$\|P - M(D, z_0 \to \cdots \to z_m)\|_\infty \leqslant \|P - M(D, z_0 \to \cdots \to z_m)\|_{\mathrm{F}} \leqslant \varepsilon = 10^{-(p+1)}$$

d'où le résultat d'après le lemme 8.8. □

8.4 Étape élémentaire de prolongement

Nous en venons maintenant au calcul de chacune des matrices M_ℓ correspondant à *un pas* de prolongement analytique. On considère toujours l'équation différentielle

$$L(z, \partial)\cdot y(z) = a_r(z)\, y^{(r)}(z) + \cdots + a_1(z)\, y'(z) + a_0\, y(z) = 0, \qquad a_k \in \mathbb{K}[z]. \tag{8.12}$$

Étant donnés deux points $z_0, z_1 \in \mathbb{K}$ suffisamment proches, il s'agit de déterminer la matrice $M(z_0 \to z_1)$ à précision ε en temps $\tilde{O}(\log(\varepsilon^{-1}))$.

L'idée générale consiste à partir de l'écriture explicite (8.8) et à évaluer chacun des coefficients de la matrice en sommant une série par scindage binaire. L'algorithme 8.12 calcule en fait tous les coefficients en une seule fois. À nouveau, quelques points méritent d'être développés, notamment la manière dont on partage une partie de l'effort entre le calcul des dérivées successives.

8.4.1 Récurrence sur le développement local

Soit y une solution de (8.12). Fixons $z_0 \in \mathbb{K}$, et posons

$$u(z) = y(z_0 + z) = \sum_{n=0}^{\infty} u_n\, z^n.$$

La suite $(u_n)_{n\in\mathbb{Z}}$ satisfait la récurrence $R\cdot u = L(z_0 + S^{-1},\, S\,n)\cdot u = 0$ d'après la proposition 3.7(a) page 48.

PROPOSITION 8.13. L'ordre s de R est le même pour tout z_0 point *ordinaire* de D, et satisfait alors

$$r \leqslant s = r + \max\{t \in \mathbb{Z} : \exists i, a_{i,i+t} \neq 0\} \leqslant r + d,$$

où l'on a posé $a_i = \sum_{j=0}^{d} a_{i,j}\, z^j$. ♢

DÉMONSTRATION. En développant $(z_0 + S^{-1})^j$ par la formule du binôme et en effectuant les commutations $S^{-j+k}\,(S\,n)^i = (n+i-j+k)^{\downarrow i}\, S^{i-j+k}$, on écrit

$$R = \sum_{i=0}^{r}\sum_{j=0}^{d}\sum_{k=0}^{j} a_{i,j}\binom{j}{k} z_0^k\,(n+i-j+k)^{\downarrow i}\, S^{i-j+k} = \sum_{i=-d}^{r} R_i(n)\, S^i.$$

Algorithme 8.12. StepTransitionMatrix$(D, z_0, z_1, \varepsilon)$

Entrée. Un opérateur différentiel $D = L(z, \partial) \in \mathbb{K}[z]\langle\partial\rangle$ d'ordre r, deux points z_0, $z_1 \in \mathbb{K}$ tels que $|z_0 - z_1| < \operatorname{dist}(z_0, \operatorname{Sing}(D))$, une précision cible $\varepsilon > 0$.

Sortie. Une matrice $\tilde{M}$ telle que $\|\tilde{M} - M(D, z_0 \to z_1)\|_{\mathrm{F}} \leqslant \varepsilon$.

Note. On suppose précalculée une série g telle que $y[z_0, j] \trianglelefteqslant g$ pour $0 \leqslant j < r$.

- *[préparer une récurrence matricielle qui fournit les sommes partielles]*

1 $\sum_{k=-s'}^{r} R_k(n)\, S^k := L(z_0 + S^{-1}, S\, n) \in \mathbb{K}[n]\langle S, S^{-1}\rangle$

2 $C := \dfrac{1}{R_r} \begin{pmatrix} & 1 & & \mathbf{0} \\ \mathbf{0} & & \ddots & \\ & & & 1 \\ R_{r-1} & R_{r-2} & \cdots & R_{-s'} \end{pmatrix} \in \mathbb{K}(n)^{s \times s}$

3 $B(n) := (\Delta, d, q, p, \Lambda)$ où

$$\begin{cases} \Delta = \operatorname{Numer}(C) & \in \mathbb{A}[n]^{s \times s} \\ d = \operatorname{Denom}(C) & \in \mathbb{Z}[n] \\ q = \operatorname{Denom}(z_1 - z_0) & \in \mathbb{Z} \\ p = \operatorname{Numer}(z_1 - z_0) + q\,\delta & \in \mathbb{A}[[\delta]]/\langle\delta^r\rangle \\ \Lambda = (\underbrace{0, \ldots, 0}_{s' \text{ fois}}, d\,q, \underbrace{0, \ldots, 0}_{(r-1) \text{ fois}}) & \in (\mathbb{A}[n][[\delta]]/\langle\delta^r\rangle)^{1 \times s} \end{cases}$$

- *[trouver un ordre de troncature convenable]*

4 calculer $N \in \mathbb{N}$ tel que $\forall i \in [\![0, r-1]\!], (g^{(i)})_{N-i;}(|z_1 - z_0|) \leqslant {}^{i!\varepsilon}/_{r}$

- *[calculer la matrice donnant la N-ième somme partielle, l'appliquer aux conditions initiales, extraire le résultat]*

5 calculer en formant un arbre de produits

$$(D, d, q, p, L) := B(N-1)\, B(N-2) \cdots B(0)$$

où chaque produit intermédiaire $P = (\Delta, d, q, p, \Lambda) = P_{\text{haut}}\, P_{\text{bas}}$ se décompose en

$$\begin{cases} \Delta := \Delta_{\text{haut}}\, \Delta_{\text{bas}} & \in \mathbb{A}^{s \times s} \\ d := d_{\text{haut}}\, d_{\text{bas}} & \in \mathbb{Z} \\ q := q_{\text{haut}}\, q_{\text{bas}} & \in \mathbb{Z} \\ p := p_{\text{haut}}\, p_{\text{bas}} & \in \mathbb{A}[[\delta]]/\langle\delta^r\rangle \\ \Lambda := p_{\text{bas}}\, (\Lambda_{\text{haut}}\, \Delta_{\text{bas}}) + d_{\text{haut}}\, q_{\text{haut}}\, \Lambda_{\text{bas}} & \in (\mathbb{A}[[\delta]]/\langle\delta^r\rangle)^{1 \times s} \end{cases}$$

6 $(V_0, \ldots, V_r) := \Lambda\, \boldsymbol{U}_0 \in \big(\mathbb{A}[[\delta]]/\langle\delta^r\rangle\big)^{1 \times s}$ où $\boldsymbol{U}_0 = \begin{pmatrix} \mathbf{0}_{s' \times r} \\ \mathbf{Id}_r \end{pmatrix} \in \mathbb{A}^{s \times r}$

7 renvoyer le quotient non simplifié $\tilde{M} := \dfrac{1}{q\,d} \Big([\delta^i]\, V_j \Big)_{0 \leqslant i, j < r} \in \mathbb{K}^{r \times r}$ ♢

Les termes de la triple somme contribuant à R_r sont exactement ceux avec $i = r$ et $j = k$: lorsque z_0 est un point ordinaire, on a donc $R_r(n) = a_r(z_0)\, (n+r)^{\downarrow r} \neq 0$.

Il est clair que si $a_{i,j} = 0$ pour $i - j \leqslant m$, alors $R_m = R_{m-1} = \cdots = 0$. Montrons par récurrence sur $m \in \mathbb{Z}$ qu'inversement, si $R_i = 0$ pour tout $i \leqslant m$, alors on a $a_{i,j} = 0$ pour $i - j = m$. Il n'y a rien à prouver pour $m < -d$. Soit $m \in \mathbb{Z}$, et supposons

$$R_m = R_{m-1} = \cdots = 0 \quad \text{et} \quad i - j \leqslant m - 1 \Rightarrow a_{i,j} = 0$$

Les termes qui contribuent à R_m sont alors ceux avec $i - j = m$ et $k = 0$, d'où

$$R_m = \sum_{i-j=m} a_{i,j}\, (n+m)^{\downarrow i}. \tag{8.13}$$

Comme les $(n+m)^{\downarrow i}$ sont des polynômes en n de degrés distincts, l'hypothèse $R_m = 0$ entraîne que tous les $a_{i,j}$ correspondants sont nuls.

Si finalement m est le plus petit indice tel que $R_m \neq 0$, l'expression (8.13) tient toujours, et donc R_m ne dépend pas de z_0. □

Notons donc s l'ordre de l'opérateur R, et posons $s' = s - r$. Soit $C(n) \in \mathbb{K}[n]^{s\times s}$ la matrice de la récurrence $R \cdot u = 0$, définie de sorte que

$$U_{n+1} = C(n)\, U_n, \qquad U_n = {}^t(u_{n-s'}, u_{n-s'+1}, ..., u_{n+r-1}). \tag{8.14}$$

Les deux premières étapes de l'algorithme 8.12 calculent R et $C(n)$. On est alors en position d'appliquer l'algorithme de la section 7.3.3 afin d'évaluer u en un point quelconque de son disque de convergence. En répétant la procédure pour chacune des solutions canoniques, on pourrait déterminer la première ligne de $M(z_0 \to z_1)$.

8.4.2 Calcul simultané des dérivées

Pour obtenir la matrice $M(z_0 \to z_1)$, toutefois, on doit évaluer en $z_1 - z_0$ non seulement u mais aussi ses quelques premières dérivées. L'article de Chudnovsky et Chudnovsky [57] considère le cas d'un système différentiel du premier ordre, pour lequel le problème ne se pose pas. S'y ramener depuis une équation scalaire est inutilement coûteux. La dérivée d'une fonction D-finie étant D-finie, on pourrait en revanche calculer des récurrences sur les coefficients de Taylor de $u', ..., u^{(r-1)}$ et mener r calculs par scindage binaire séparés.

Van der Hoeven [235, §4.1] propose comme alternative d'introduire une perturbation formelle δ du point d'évaluation et de calculer par scindage binaire le développement en série tronqué

$$y(z_1+\delta) \bmod \delta^r = y(z_1) + y'(z_1)\,\delta + \cdots + \frac{y^{(r-1)}(z_1)}{(r-1)!}\,\delta^{r-1} \in \mathbb{K}[[\delta]]/\langle\delta^r\rangle.$$

Il poursuit en calculant une récurrence dépendant de $z = z_1 - z_0 + \delta$ satisfaite par la suite $(u_n\, z^n)$, et ce faisant, n'exploite pas pleinement cette idée.

On gagne en effet en efficacité en la combinant à une variante de la technique de la section 7.3.3. Au lieu de passer par une récurrence sur le produit $u_n\, z^n$, on calcule u_n par la relation (8.14) et, parallèlement, la puissance $z^n \in \mathbb{K}[[\delta]]/\langle\delta^r\rangle$. On complète la matrice de la récurrence pour obtenir les sommes partielles en plus des coefficients comme dans le cas des simples sommes de séries numériques. Ainsi, seuls le calcul de z^n et l'accumulation des sommes partielles opèrent sur des séries en δ. Celui de U_n, seule étape à faire intervenir un produit matriciel en taille s, est partagé entre les dérivées $u, ..., u^{(r-1)}$.

Précisons un peu. Visuellement, le système du premier ordre donnant les sommes partielles est donc

$$\begin{pmatrix} u_{n-s'+1}\, z^{n+1} \\ \vdots \\ u_n\, z^{n+1} \\ u_{n+1}\, z^{n+1} \\ \vdots \\ u_{n+r-1}\, z^{n+1} \\ u_{n+r}\, z^{n+1} \\ u_{;n+1}(z) \end{pmatrix} = \left(\begin{array}{cccccccc|c} 1 & & & & & & & & 0 \\ & \ddots & & & & \mathbf{0} & & & \\ & & 1 & & & & & & \\ & & & 1 & & & & & z \\ & \mathbf{0} & & & \ddots & & & & \vdots \\ & & & & & & 1 & & \\ * & * & \cdots & * & * & \cdots & * & & 0 \\ \hline 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & & 1 \end{array}\right) \begin{pmatrix} u_{n-s'}\, z^n \\ \vdots \\ u_{n-1}\, z^n \\ u_n\, z^n \\ \vdots \\ u_{n+r-2}\, z^n \\ u_{n+r-1}\, z^n \\ u_{;n}(z) \end{pmatrix} \tag{8.15}$$

et il s'agit de calculer le N-ième terme de la suite récurrente ainsi définie par scindage binaire, en veillant au cours du calcul à garder factorisé le produit $C(n)\, z$ dans le bloc en haut à gauche. Après réduction au même dénominateur de tous les coefficients, le numérateur de la matrice de l'équation (8.15) s'écrit

$$B(n) = \begin{pmatrix} \Delta(n)\, p & 0 \\ \Lambda & d\, q \end{pmatrix} = d\, q \begin{pmatrix} C(n)\, z & 0 \\ K & 1 \end{pmatrix} \tag{8.16}$$

(où K désigne le vecteur $(0, ..., 0, 1, 0, ..., 0)$, avec le 1 en $(s'+1)$-ième position), forme calculée à l'étape 3 de l'algorithme 8.12.

Remarque 8.14. On pourrait attendre $d \in \mathbb{A}[n]$ plutôt que $d \in \mathbb{Z}[n]$. Mais on a $R_r(n) = a_r(z_0)\,(n+r)^{\downarrow r}$ (voir la preuve de la proposition 8.13), ce qui permet de se ramener à un dénominateur à coefficients rationnels en multipliant par le conjugué de $a_r(z_0)$. Pour un point singulier régulier — où R_r s'annule mais le coefficient de tête de la récurrence reste de degré r en n — la propriété analogue ne tient plus. Par exemple, si z_0 est racine simple de a_r, on trouve

$$R_{r-1} = a_r'(z_0)\,(n+r-1)^{\downarrow(r)} + a_{r-1}(z_0)\,(n+r-1)^{\downarrow(r-1)}$$

qui n'est pas en général dans $\mathbb{K}\cdot\mathbb{Q}[n]$. ♢

Remarque 8.15. Autre conséquence de la forme de $R_r(n)$: on connaît explicitement la décomposition en facteurs premiers du dénominateur, par la formule de Legendre

$$n! = \prod_{p\text{ premier}} p^{v_p(n!)}, \qquad v_p(n!) = \sum_{k=0}^{\infty} \left\lfloor \frac{n}{p^k} \right\rfloor.$$

Cela permet de le calculer plus efficacement que par scindage binaire [32, 218, 162]. Il pourrait aussi être intéressant d'utiliser cette factorisation pour éliminer les diviseurs communs entre numérateur et dénominateur sans recourir à l'algorithme général de calcul de pgcd. ♢

Un produit $P(i,j) = B(j-1)\cdots B(i)$ de matrices (8.16) est lui-même de la forme

$$B(j-1)\cdots B(i) = \begin{pmatrix} \Delta\, p & 0 \\ \Lambda & d\, q \end{pmatrix}.$$

La multiplication

$$\begin{pmatrix} \Delta\, p & 0 \\ \Lambda & d\, q \end{pmatrix} = \begin{pmatrix} \Delta_{\text{haut}}\, p_{\text{haut}} & 0 \\ \Lambda_{\text{haut}} & d_{\text{haut}}\, q_{\text{haut}} \end{pmatrix} \begin{pmatrix} \Delta_{\text{bas}}\, p_{\text{bas}} & 0 \\ \Lambda_{\text{bas}} & d_{\text{bas}}\, q_{\text{bas}} \end{pmatrix},$$

s'effectue par les formules données à l'étape 5 de l'algorithme.

Une analyse de complexité détaillée serait un peu laborieuse, mais on peut proposer l'estimation heuristique suivante.

Estimation 8.16. (Cette analyse n'a aucune prétention à la rigueur.) L'opérateur R est d'ordre s, de degré r, et de hauteur de l'ordre de $k\,\mathrm{haut}(z_0)$ où k est le degré en z de $L(z,\partial)$. Dans le cas des opérateurs de récurrence qui annulent les suites de coefficients des dérivées u', ..., $u^{(r-1)}$, le degré monte à environ s. Passer d'un opérateur d'ordre s annulant (u_n) à un opérateur annulant $(u_n\, z^n)$ fait augmenter la hauteur d'environ $s\,\mathrm{haut}(z)$. Ainsi, si L est de hauteur négligeable et $\mathrm{haut}(z_0) + \mathrm{haut}(z_1) \leqslant h$, la récurrence sur les coefficients $u_n^{(i)}\,(z_1 - z_0)^n$ que l'on utiliserait pour calculer « directement » $y^{(i)}(z_1)$ est d'ordre égal à s, de degré de l'ordre de s et de hauteur de l'ordre de $s\,h$. Le coût du calcul d'une somme partielle par scindage binaire correspondant est alors de l'ordre de

$$\mu(s)\, M(s\,N\,(h + \lg N))\, \lg N$$

où $\mu(s) = O(s^\omega)$ désigne le nombre de multiplications dans $\mathbb{Z}$ pour chaque produit de matrices $s \times s$. Ce coût est à multiplier par r pour obtenir $y(z_1), ..., y^{(r-1)}(z_1)$.

En comparaison, le coût du produit de matrices de la forme (8.16) tel qu'effectué dans l'algorithme se ramène essentiellement à celui de la multiplication $\Delta_{\text{haut}}\,\Delta_{\text{bas}}$. En effet, la multiplication $\Lambda_{\text{haut}}\,\Delta_{\text{bas}}$ peut se réécrire comme un produit de matrices $r \times s$ par $s \times s$ (or $r \leqslant s$), les s produits dans $\mathbb{A}[[\delta]]/\langle\delta^r\rangle$ pour multiplier le résultat par p_{bas} se font chacun en $M_{\mathbb{A}}(r)$ opérations dans $\mathbb{A}$, et même en r opérations d'après des remarques à venir en §7.4. Nous verrons aussi en §7.4 que l'on peut sous certaines hypothèses prendre $\mu(s) = O(s^2)$, mais même dans ce cas, ces opérations (qui, par

ailleurs, bénéficient aussi des techniques qui font descendre $\mu(s)$) restent négligeables. Le coût total du scindage binaire dans ces conditions est de l'ordre de

$$\mu(s)\, M(N\,(k\,h + r \lg N)) \lg N$$

soit environ autant que pour le calcul d'une seule dérivée. Cela suggère un gain total d'un facteur de l'ordre de r, voire un peu plus si les rapports

$$\frac{\mathrm{haut}(z_1)}{\mathrm{haut}(z_0)}, \quad \frac{s}{r}, \quad \frac{s}{\deg_z L(z,\partial)}$$

sont favorables. ♢

EXEMPLE 8.17. Reprenons l'équation différentielle aléatoire d'ordre 4 de la page 25, et calculons la valeur de la solution y au point $\frac{1}{4}$ à précision 10^{-2000}. NumGfun effectue le calcul en une seule étape de prolongement analytique, en 4,5 secondes environ. Le calcul de la matrice de transition complète $M\big(0 \to \frac{1}{4}\big)$ prend 5,4 secondes, soit nettement moins de quatre fois plus ! ♢

8.4.3 Conditions initiales et parcours des solutions canoniques

Nous venons de voir comment calculer en une fois une colonne de la matrice de transition. Naturellement, l'opérateur R ne dépend pas du choix de $y \in \ker D$. Ainsi, les coefficients d'une même ligne de la matrice $M(z_0 \to z_1)$, c'est-à-dire les valeurs en z_1 d'une même dérivée des différentes solutions canoniques $y_{[z_0,j]}$, s'obtiennent en ne changeant que les conditions initiales de la récurrence.

L'ordre de la récurrence pouvant être plus grand que celui de l'équation différentielle, on est conduit à « compléter » les conditions initiales

$$(u_0, ..., u_{r-1}) = (0, ..., 0, 1, 0, ..., 0)$$

qui définissent chaque solution fondamentale en un vecteur de taille $s \geqslant r$. Comme la relation de récurrence $R \cdot u = 0$ est valable sur $\mathbb{Z}$, il suffit d'ajouter les conditions $u_n = 0$ pour $n < 0$. Avec la convention (8.14), la matrice de conditions initiales

$$\boldsymbol{U}_0 := \begin{pmatrix} \mathbf{0}_{s' \times r} \\ \boldsymbol{I}_r \end{pmatrix} = \big(\; [z^i]\, y_{[z_0,j]}(z_0 + z) \; \big)_{\substack{-s' \leqslant i < r \\ 0 \leqslant j < r}}$$

qui apparaît ligne 6 de l'algorithme 8.12 est telle que

$$B(n-1) \cdots B(0) \begin{pmatrix} \boldsymbol{U}_0 \\ 0 \end{pmatrix} = q\,d \begin{pmatrix} * & \cdots & * \\ \vdots & & \vdots \\ * & \cdots & * \\ y_{[z_0,0]}(z_1 + \delta) & \cdots & y_{[z_r,0]}(z_1 + \delta) \end{pmatrix} \pmod{\delta^r}$$

où $q\,d$ désigne le coefficient en bas à droite de $B(n-1) \cdots B(0)$.

8.4.4 Contrôle de l'erreur locale

Il reste enfin à contrôler l'erreur sur $M(z_0 \to z_1)$ introduite en tronquant les séries qui donnent les coefficients. C'est l'objet de l'étape 4 de l'algorithme 8.12.

Étant donnée une borne de précision ε (qui provient de l'étape de contrôle d'erreur global de l'algorithme 8.5, voir §8.3.3), notre mission est simplement de choisir un ordre de troncature N tel que

$$\|\tilde{M} - M(D, z_0 \to z_1)\|_{\mathrm{F}} \leqslant \varepsilon.$$

Concernant les troncatures des dérivées, on a

$$u_{;n}(z + \delta) = \sum_{i=0}^{r-1} \frac{1}{i!} (u_{;n})^{(i)}(z) = \sum_{i=0}^{r-1} \frac{1}{i!} \big(u^{(i)}\big)_{;n-i}(z) \pmod{\delta^r}.$$

Comme en §8.3.3, on se sert d'une série majorante g commune à toutes les solutions fondamentales. On recherche par doublements successifs de n puis par dichotomie le plus petit n pour lequel on peut certifier

$$\frac{1}{i!}\left(g^{(i)}\right)_{n-i;}(|z_1-z_0|)\leqslant\frac{\varepsilon}{r}\qquad\text{pour tout } i\in[\![0,r-1]\!],$$

ce qui entraîne

$$\|\tilde{M}-M(D,z_0\to z_1)\|_{\mathrm{F}}\leqslant\left\|G_{;n}\left(\left|z_1-z_0\right|\right)\right\|_{\mathrm{F}}\leqslant r\left\|G_{;n}\left(\left|z_1-z_0\right|\right)\right\|_{\infty}\leqslant\varepsilon.$$

Les bornes sur les restes donnés par la proposition 6.5 ne contiennent que des fonctions élémentaires (plus la fonction Γ en des points rationnels, que l'on peut au besoin remplacer par une borne en termes de $n!$ au moyen du lemme 5.35), faciles à évaluer en arithmétique d'intervalles.

8.5 Cas d'un point d'évaluation donné à grande précision

8.5.1 Algorithme *bit burst*

La complexité obtenue dans la proposition 8.2 est quasi-linéaire en la précision p lorsque la hauteur h du point d'évaluation croît au plus comme $O(\lg p)$. En revanche, elle devient quadratique si $h=\Theta(p)$. Or, considérons l'évaluation d'une fonction y en un point $z\in\mathbb{C}\setminus\mathbb{K}$. On peut calculer une valeur approchée de $y(z)$ en remplaçant z par un point $\tilde{z}\in\mathbb{K}$ suffisamment proche, mais on s'attend en faisant cela à introduire une erreur de l'ordre de $|f'(z)\,(z-\tilde{z})|$. Pour calculer $y(z)$ à précision 2^{-p}, il faut donc en général prendre une approximation $\tilde{z}$ de taille $\Omega(p)$.

On contourne cette difficulté en calculant $y(\tilde{z})$ par prolongement analytique le long d'un chemin

$$z_0\to z_1\to\cdots\to z_m=\tilde{z}$$

formé d'approximations de z dont la précision croît géométriquement. Adoptons pour z_ℓ la troncature du développement binaire de z à $2^\ell+1$ chiffres après la virgule. On a alors

$$|z_{\ell+1}-z_\ell|\leqslant 2^{-2^\ell}\qquad\text{et}\qquad\operatorname{haut}(z_\ell)=O(2^\ell),$$

ce qui équilibre les contributions de $|z|$ et h dans (8.1).

Des cas particuliers de cet algorithme remontent à l'article fondateur de Brent [37] sur l'évaluation à grande précision des fonctions élémentaires. L'idée dans le cas général est due à Chudnovsky et Chudnovsky [57, 56], puis indépendamment à van der Hoeven [235, 241] avec une analyse de complexité un peu plus fine. La proposition suivante améliore cette dernière d'un facteur $\lg\lg p$ dans le cas où le rayon de convergence de y est fini, sans changement à l'algorithme.

Proposition 8.18. Soit y une fonction D-finie. On peut calculer une approximation à 2^{-p} près de $y(z)$ en $O(M(p\lg^2 p))$ opérations binaires lorsque $p\to\infty$ et z varie parmi les éléments de $\mathbb{K}$ de hauteur bornée par un multiple constant de p. ♢

Démonstration. Tout d'abord, $m=O(\lg\,p)$ étapes de prolongement analytique suffisent. D'après la proposition 8.2, le pas $z_\ell\to z_{\ell+1}$ demande

$$O\left(M\left(\frac{p\,(2^\ell+\lg p)}{2^\ell}\right)(m+\lg p)\right)=O\left(M\left(p+\frac{\lg p}{2^\ell}\right)\lg p\right)$$

opérations binaires. On a par super-linéarité de la multiplication

$$\sum_{\ell=0}^{m}M\left(p+\frac{p\lg p}{2^\ell}\right)\lg p\leqslant M\left(m\,p+\sum_{\ell=0}^{\infty}\frac{p\lg p}{2^\ell}\right)\lg p$$

donc le coût total du calcul est de $O(M(p \lg p) \lg p)$ opérations binaires. □

Cela unifie la borne de complexité avec celle pour les fonctions entières, pour lesquelles notre analyse n'apporte pas d'amélioration[4].

EXEMPLE 8.19. La sortie ci-dessous retrace le déroulement du calcul par l'algorithme *bit-burst* de $\arctan(e)$ effectué dans le premier exemple de la section 1.2.5, page 28. Les messages d'information donnés par NumGfun ont été légèrement simplifiés par souci de lisibilité. Pour chaque pas de prolongement analytique sont indiqués les extrémités du pas, la précision de calcul de la matrice de transition, le nombre de lignes de celles-ci calculées, et le nombre de termes de la série correspondante qu'il a fallu sommer.

```
> my_e := convert(evalf[5000](exp(1)), rational, exact):
> diffeq := holexprtodiffeq(arctan(z), y(z)):
> infolevel[gfun] := 2:
> evaldiffeq(diffeq, y(z), [0,my_e], 5000);
   ~2.718281828 --> ~2.718281828 ord=1, prec~=.14251e-5006, terms=4
   0 --> 1/2 ord=2, prec~=.27681e-5006, terms=16690
   1/2 --> 11/10 ord=2, prec~=.20112e-5008, terms=18596
   11/10 --> 9/5 ord=2, prec~=.90600e-5009, terms=15366
   9/5 --> 34512587/12696500 ord=2, prec~=.20112e-5008, terms=14330
   34512587/12696500 --> 157748906311/58032579500 ord=2, prec~=.85970e-5010, terms=886
   157748906311/58032579500 --> ~2.718281828 ord=2, prec~=.17194e-5009, terms=464
   ~2.718281828 --> ~2.718281828 ord=2, prec~=.70017e-5009, terms=248
   ~2.718281828 --> ~2.718281828 ord=2, prec~=.85970e-5010, terms=124
   ~2.718281828 --> ~2.718281828 ord=2, prec~=.85970e-5010, terms=62
   ~2.718281828 --> ~2.718281828 ord=2, prec~=.85970e-5010, terms=32
   ~2.718281828 --> ~2.718281828 ord=2, prec~=.85970e-5010, terms=16
   ~2.718281828 --> ~2.718281828 ord=2, prec~=.85970e-5010, terms=10
   ~2.718281828 --> ~2.718281828 ord=2, prec~=.85970e-5010, terms=6
   1.21828290501727762176...(4660 chiffres)...85796212560201267299
```

On observe la croissance des tailles en bits des points d'évaluation et leur convergence vers e, et parallèlement la décroissance géométrique du nombre de termes à sommer. La précision cible reste quant à elle à peu près constante d'une étape à l'autre. ♢

8.5.2 Majoration des solutions canoniques en un point de grande taille

Une seconde difficulté se présente lorsque la hauteur d'un point z_0 du chemin de prolongement analytique — et donc de l'équation différentielle translatée correspondante — est grande. Le calcul de la série majorante g telle que

$$\forall j, \quad y_{[z_0,j]}(z_0 + z) \trianglelefteq g(z)$$

qui aboutit à la borne (8.10) elle-même à la base du contrôle des erreurs devient extrêmement coûteux. On contourne ce problème en bornant les solutions canoniques en z_0 à partir de l'équation différentielle prise en un point $c \approx z_0$. Dans les phases de *bit burst*, on réutilise le même point c pour plusieurs pas de prolongement, ce qui évite de multiplier les calculs de majorants.

4. Grossièrement, si l'on considère un chemin $z_0 \to z_1 \to \cdots \to z_m$ où z_i est la troncature à h_i bits de $z = z_m$, on a $|z_{i+1} - z_i| \approx 2^{-h_i}$, d'où $N_i (\lg N_i - \lg |z_{i+1} - z_i|) \approx N_i (\lg N_i + h_i) \approx p$ si l'équation différentielle n'a pas de point singulier fini. Le coût du prolongement analytique le long du chemin est de l'ordre de

$$\sum_{i=0}^{m-1} M(N_i (\lg N_i + h_{i+1}))\, (\lg N_i + \lg h_{i+1}) \approx \sum_{i=0}^{m-1} M\Big(\frac{h_{i+1}}{h_i}\, p \lg p\Big)$$

qui est minimal et de l'ordre de $M(p \lg^2 p)$ pour $h_{i+1}/h_i = \Theta(1)$. On voit en passant que le découpage $h_{i+1} = 2\ h_i$ est essentiellement optimal. Le même argument fonctionne pour un rayon de convergence fini : il s'ajoute un terme de l'ordre de $\sum_i M((\lg p/h_i)\, p \lg p)$ qui est $O(M(p \lg^2 p))$ pour le choix de h_i qui minimise l'autre terme.

Proposition 8.20. Soit $c \in \mathbb{C}$. Soit $G \in \mathbb{R}_+[[z]]^{r\times r}$ un majorant de la matrice fondamentale canonique au point c de l'équation (8.12), en symboles,

$$\boldsymbol{Y}[c](c+z) \preccurlyeq G(z).$$

Soit η tel que $0 < \eta < \operatorname{dist}(c, \operatorname{Sing}(D))$ et

$$\alpha = \|G(\eta) - G(0)\|_{\mathrm{F}} < 1.$$

Alors on a pour tout $z_0 \in \mathcal{D}(c,\eta)$ la majoration

$$\boldsymbol{Y}[z_0](z_0+z) \preccurlyeq \frac{G(\eta+z)\,\mathbf{1}}{1-\alpha}$$

où $\mathbf{1}$ représente la matrice $r\times r$ dont tous les coefficients sont égaux à 1. ♢

Démonstration. Soit z_0 tel que $|z_0 - c| < \eta$. La règle de prolongement (8.9) couplée à celle de composition des séries majorantes (lemme 5.18) entraîne

$$\begin{aligned}\boldsymbol{Y}[z_0](z_0+z) &= \boldsymbol{Y}[c]\big(c+z+(z_0-c)\big)\, M(D, c\to z_0)^{-1}\\ &\preccurlyeq G\big(|z_0-c|+z\big)\, \|M(D,c\to z_0)^{-1}\|_{\mathrm{F}}\,\mathbf{1}.\end{aligned}$$

Par définition des solutions canoniques, le terme constant de la série $M(D, c\to c+z)$ est la matrice identité, donc

$$M(D,c\to c+z) - \mathrm{Id} \preccurlyeq G(z) - G(0).$$

et $\|M(D,c\to c+z) - \mathrm{Id}\|_{\mathrm{F}} \leqslant \alpha < 1$. On en déduit

$$\|M(D,c\to c+z)^{-1}\|_{\mathrm{F}} \leqslant \frac{1}{1-\alpha}$$

et le résultat. □

Si $g(z) \in \mathbb{R}_+[[z]]$ est un majorant commun des solutions canoniques, on peut prendre (cf. équation (8.10))

$$G(z) = \Big(\tfrac{1}{i!}\, g^{(i)}(z) \Big)_{0\leqslant i,j<r}$$

et on obtient

$$y[z_0, j](z) \preccurlyeq \frac{r}{1-\alpha}\, g(\eta+z)$$

pour tout j. Nous avons vu comment majorer les valeurs et les restes de $y[z_0, j]$ à partir de cette inégalité (remarque 6.7).

Observons aussi qu'une équation aussi simple que $(1-z)^2\, y(z) = K\, y(z)$ a pour solution la fonction $z \mapsto \exp(K/(1-z))$ où K peut être très grand. La condition $\alpha < 1$ de la proposition 8.20 impose de prendre c de hauteur $\Omega(K)$. Dans l'implémentation de cette méthode dans NumGfun, on prend initialement pour c une approximation à petite précision de z_0, et si l'inégalité $\alpha < 1$ n'est pas satisfaite, on se sert de la borne trouvée pour réessayer avec une approximation dont on a augmenté la précision d'environ K, jusqu'à avoir $\alpha < 1$ ou $c = z_0$.

Ce genre d'équation est de toute façon difficile à traiter avec les algorithmes de ce chapitre, voire plus généralement. Il s'agit en effet d'approcher avec une erreur absolue petite une fonction aux variations très rapides sur le domaine considéré.

Remarque 8.21. Van der Hoeven [235, §4.3] fait un raisonnement similaire à la preuve de la proposition 8.20 pour choisir automatiquement à quelle précision approcher un point z_1 lui-même donné par un programme d'évaluation à précision arbitraire par un point voisin $\tilde{z}_1 \in \mathbb{K}$ pour garantir $|y(\tilde{z}_1) - y(z_1)| \leqslant \varepsilon$ pour ε donné. Cette possibilité n'est pas implémentée dans NumGfun, qui se limite à l'évaluation en des points à coordonnées rationnelles. ♢

8.6 Connexion aux points singuliers réguliers

8.6.1 Prolongement analytique à travers un point singulier

Nous avons vu au chapitre 4 qu'au voisinage d'un point singulier régulier z_0, l'équation différentielle (8.12) admet une base de solutions de la forme

$$y(z) = (z-z_0)^\lambda \left(\varphi_0(z) + \varphi_1(z)\log(z-z_0) + \cdots + \varphi_{t-1}(z)\log^{t-1}(z-z_0)\right)$$

où les φ_k sont des fonctions *analytiques* au voisinage de z_0. Les coefficients du développement en série entière en z_0 de ces dernières satisfont un système de récurrences. Connaissant la décomposition d'une fonction sur cette base, on peut donc l'évaluer en des points suffisamment proches de z_0. Mieux, on peut calculer la matrice qui fait passer des coefficients de cette décomposition aux conditions initiales de l'équation différentielle en un point ordinaire voisin, et généraliser à ce cas l'algorithme de prolongement analytique de la section 8.3.

Solutions canoniques et détermination du logarithme. Plaçons-nous pour alléger les notations en $z_0 = 0$, que l'on suppose être un point régulier, c'est-à-dire ordinaire ou singulier régulier. On adopte la base de solutions canoniques définie en §4.4.3, à savoir les séries logarithmiques

$$y[0,e] = \sum_{n\in\lambda+\mathbb{Z}} \sum_{k\in\mathbb{N}} y[0,e]_{n,k} \frac{\log^k z}{k!} z^n, \qquad e\in E = \{(n,k) : 0 \leqslant k < \mu(n)\} \tag{8.17}$$

caractérisées par

$$\begin{cases} y[0,(n,k)]_{n,k} = 1 \\ y[0,(n,k)]_{n',k'} = 0 \quad \text{pour } (n',k') \in E \setminus \{(n,k)\}. \end{cases}$$

À ce stade, ce sont des séries formelles. Pour tout k, la série $\sum_n y[0,e]_{n,k}\, z^{n-\lambda}$ est convergente sur un certain disque $\mathcal{D}(0,\rho)$, $\rho > 0$; mais pour donner un sens à l'évaluation de $y[0,e]$ elle-même en un point $z_1 \in \mathcal{D}(0,\rho)$, il reste à spécifier à quelle branche des « fonctions multivaluées » $\log z$ ou z^λ (si $\lambda \notin \mathbb{Z}$) les éléments $\log z_1$ ou z_1^λ qui peuvent apparaître font référence. Il suffit pour cela de choisir comment s'interprète le logarithme : z^λ est alors défini sans ambiguïté par $z^\lambda = e^{\lambda \log z}$.

La convention adoptée dans NumGfun, et dans ce document, est que le symbole $\log$ désigne la détermination standard du logarithme (voir page 45). On considère donc les développements des solutions de $D\cdot y = 0$ comme des fonctions analytiques sur le disque fendu $\mathcal{D}(0,\rho)\setminus\mathbb{R}_-$. Le transfert de (8.17) à d'autres points $z_0 \in \mathbb{C}$ se faisant par translation (*i.e.* par le changement de variable $z \mapsto z - z_0$, plutôt que, disons, $z \mapsto z/z_0 - 1$), la coupure utilisée pour définir le logarithme pointe dans la direction des négatifs au voisinage de toute singularité finie.

Prolongement en une singularité. On donne ainsi un sens au « prolongement analytique » d'une solution le long d'un chemin pouvant passer par un point singulier régulier. Limitons-nous (sans perte de généralité) comme d'habitude aux lignes brisées, avec en plus la restriction que les points singuliers ne peuvent apparaître qu'aux sommets du chemin.

Soient z_0, z_1 des points réguliers de (8.12). Soit y une combinaison linéaire formelle des solutions canoniques $y[z_0, e]$. Elle définit une fonction analytique sur le disque fendu $\mathcal{D}(z_0,\varepsilon)\setminus(z_0+\mathbb{R}_-)$, prolongée à $\mathcal{D}(z_0,\varepsilon)$ par continuité sur le quadrant nord-ouest. Le prolongement de y le long du chemin $z_0 \to z_1$, supposé ne pas contenir d'autre point singulier que z_0 et z_1, est la fonction analytique qui coïncide avec y sur un intervalle initial $]z_0, z_1[\cap\mathcal{D}(z_0,\varepsilon)$. Son prolongement en z_1 est la combinaison linéaire des $y[z_1, e']$ qui coïncide avec y (prolongée) sur un intervalle $]z_0, z_1[\cap\mathcal{D}(z_1,\varepsilon')$. On étend la définition à un chemin quelconque par concaténation.

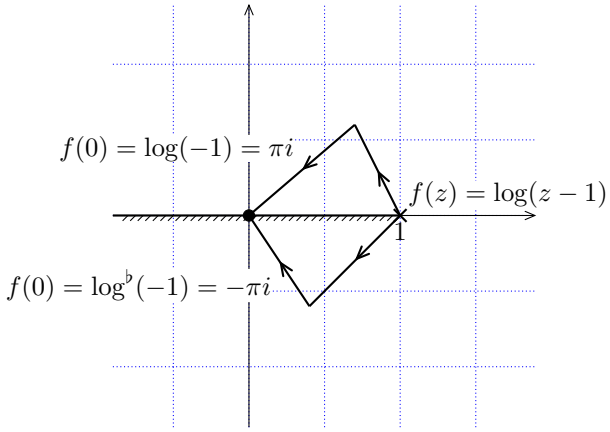


Figure 8.2. Prolongement analytique à partir d'une singularité : deux chemins homotopes dans $\mathbb{C} \setminus \mathrm{Sing}(D)$ qui conduisent à des résultats différents.

L'interprétation d'un tel chemin dépend donc de la *direction* des pas dont une des extrémités est un point singulier, comme l'illustre la figure 8.2. Concrètement, en déformant le chemin, on n'est pas libre de faire traverser aux segments qui quittent un point singulier ou y aboutissent la direction des réels négatifs. Un chemin de prolongement qui passe par un point singulier ζ et « en ressort » équivaut au chemin déformé pour contourner ζ « par la droite », sans intersecter la demi-droite $\zeta + \mathbb{R}_-$.

Remarque 8.22. On pourrait généraliser la notion de prolongement analytique à travers un point singulier pour permettre de « ressortir sur un autre feuillet de la surface de Riemann de l'équation », c'est-à-dire avec potentiellement une autre détermination des solutions. Van der Hoeven [236, §4] détaille une telle construction, utile pour énoncer des algorithmes de calcul de monodromie locale (voir §8.7) à partir des développements aux points singuliers. ♢

Matrices de transition. Afin de déterminer sans ambiguïté les matrices de transition le long de chemins dont une extrémité est un point singulier, on ordonne les éléments de E par

$$(n,k) < (n',k') \quad \Leftrightarrow \quad \begin{cases} & \mathrm{Re}\, n < \mathrm{Re}\, n' \\ \text{ou} & \mathrm{Re}\, n = \mathrm{Re}\, n' \ \text{ et } \ k > k' \\ \text{ou} & \mathrm{Re}\, n = \mathrm{Re}\, n' \ \text{ et } \ k = k' \ \text{ et } \ \mathrm{Im}\, n < \mathrm{Im}\, n'. \end{cases} \tag{8.18}$$

Cet ordre traduit le comportement asymptotique du monôme $z^n \log^k z$: sauf si les couples comparés ne diffèrent que par la partie imaginaire de n, on a

$$(n,k) < (n',k') \quad \Leftrightarrow \quad |z^n \log^k z| \gg |z^{n'} \log^{k'} z| \ \text{ quand } z \to 0.$$

Les définitions et propriétés de la section 8.3.1 s'étendent en remplaçant $[\![0, r-1]\!]$ par E aux endroits convenables. À toute solution $y \in \ker D$, on associe le vecteur colonne de conditions initiales généralisées à l'origine

$$Y(0) = {}^{\mathrm{t}}(\ y_{n,k}\ )_{(n,k)\in E} \tag{8.19}$$

(les éléments de E étant parcourus par ordre croissant). On retrouve la définition (8.4) si 0 est un point ordinaire. De même, on définit le vecteur ligne de solutions canoniques $\boldsymbol{y}[0](z) = (y[0,e](z))_{e\in E}$, et les matrices de transition le long de chemins passant par des points singuliers réguliers en remplaçant les conditions initiales classiques (8.4) par les conditions intiales généralisées (8.19) dans leur définition précédente (8.7). La matrice de transition en ligne droite de $z_0 = 0$ (régulier) à z_1 (ordinaire) est

$$M(D, z_0 \to z_1) = \boldsymbol{Y}[z_0](z_1) = \Big(\ \tfrac{1}{i!}\, y^{(i)}_{[z_0,e]}(z_1)\ \Big)_{\substack{0\leqslant i<r \\ e\in E}}. \tag{8.20}$$

Comme dans le cas de deux points ordinaires, cette expression nous permettra d'évaluer $M(D, z_0 \to z_1)$. Son inverse $M(D, z_1 \to z_0)$ ne s'exprime pas aussi simplement. Pour calculer une matrice de transition d'un point ordinaire vers un point singulier, nous passerons par la matrice inverse.

8.6.2 Matrice de transition d'un point singulier vers un point ordinaire

Comment calculer la matrice (8.20) ? Essentiellement comme dans le cas de points ordinaires : nous allons évaluer ses coefficients par scindage binaire, en se servant du fait que les suites $(y[z_0, e]_{n,k})_n$ à k fixé sont P-récursives. Une fois de plus, l'idée date des travaux des frères Chudnovsky [57, p. 125] (voir aussi [53, 55]). Ceux-ci ne traitent cependant pas les difficultés algorithmiques spécifiques au cas singulier régulier. Van der Hoeven [236] donne un premier algorithme complet. La présente version est un peu plus efficace et probablement plus simple à mettre en œuvre.

Implémenter ce schéma sans redondances grossières avec les primitives d'un logiciel de calcul formel demande tout de même un peu de soin. Je donne ici une description technique relativement précise d'une organisation possible, qui suit à quelques détails près ce qui est fait dans NumGfun. Une grande partie de cette section un peu lourde s'adresse principalement au lecteur qui souhaiterait implémenter le prolongement analytique depuis un point singulier régulier. Sur le plan théorique, le message est simplement que la récurrence obtenue en réinterprétant la méthode de Poole (proposition 4.16 p. 70) permet une extension naturelle et efficace aux développements en série généralisées du calcul par scindage binaire des sommes partielles de solutions.

L'algorithme 8.23 représente une généralisation de l'algorithme 8.12 couvrant le cas où z_0 est un point singulier régulier au lieu d'un point ordinaire. Pour étendre le prolongement analytique aux chemins passant par des points singuliers réguliers, il suffit de remplacer l'appel à StepTransitionMatrix dans l'algorithme 8.5 par un appel à SingularStepTransitionMatrix avec les mêmes paramètres. Passons en revue les principales différences entre ces deux fonctions.

Récurrence. Le point de départ de l'algorithme 8.23 est la récurrence sur les suites doubles $(y[z_0, e]_{n,k})_{n,k}$ introduite au chapitre 4. Supposons à nouveau $z_0 = 0$, et quitte à multiplier l'opérateur D par une puissance de z, changeons légèrement de notations en posant $D = L(z, \theta)$ au lieu de $D = L(z, \partial)$.

D'après la proposition 4.14 page 69 et avec les conventions d'interprétation des opérateurs S_n et S_k correspondantes, on a pour tout e la relation de récurrence

$$L(S_n^{-1}, n + S_k) \cdot (y[z_0, e]_{n,k})_{n \in \lambda + \mathbb{Z}, k \in \mathbb{N}} = 0. \tag{8.21}$$

On note s son ordre vis-à-vis de S_n. En comparaison de la situation pour un point ordinaire (proposition 8.13), on a toujours $s \leqslant d + r$ où d est le degré en z de l'opérateur $D \in \mathbb{Q}[z]\langle\partial\rangle$ initial, mais plus forcément $s \geqslant r$. Exemple typique, la fonction hypergéométrique classique $y(z) = {}_2\mathrm{F}_1(a, b\,; c; z)$ est solution de l'équation d'ordre 2

$$z\,(z-1)\,y''(z) + \big((a+b+1)\,z - c\big)\,y'(z) + a\,b\,y(z) = 0$$

alors que les coefficients y_n de son développement en série à l'origine (qui est un point singulier régulier) vérifient la récurrence du premier ordre $(n+1)\,(n+c)\,y_{n+1} = (n+a)\,(n+b)\,y_n$.

Matrice de récurrence généralisée. Comme observé lorsque nous l'avons introduite, la récurrence (8.21) se prête au scindage binaire. On y adapte la construction des matrices

$$B(n) = d\,q \begin{pmatrix} C(n)\,z & 0 \\ K & 1 \end{pmatrix}, \qquad z = z_1 - z_0 + \delta \in \mathbb{K}[[\delta]]/\langle\delta^r\rangle, \tag{8.22}$$

Algorithme 8.23. SingularStepTransitionMatrix$(D, z_0, z_1, \varepsilon)$

Entrée. Un opérateur différentiel $D \in \mathbb{Q}[z]\langle\partial\rangle$. Un point *régulier* $z_0 \in \mathbb{K}$ et un point *ordinaire* $z_1 \in \mathbb{K}$ de D avec $|z_1 - z_0| < \operatorname{dist}(z_0, \operatorname{Sing}(S))$. Une erreur cible ε.

Sortie. Une approximation $\tilde{M}$ de la matrice de transition généralisée de z_0 à z_1 telle que $\|\tilde{M} - M(D, z_0 \to z_1)\|_{\mathrm{F}} \leqslant \varepsilon$.

Notes. On suppose précalculée une série majorante $g(z) \in \mathbb{Q}_+[[z]]$ des solutions canoniques généralisées en z_0, au sens de la proposition 6.20. Tous les calculs sont faits sans simplifier les fractions, et en gardant les matrices (8.22) et leurs produits décomposés sous la forme (8.24). On note $a_0 + a_1 \operatorname{Log} + a_2 \operatorname{Log}^2 + \cdots$ les suites $(a_0, a_1, a_2, ...)$ sur lesquelles agit S_k (ainsi, $S_k \cdot \operatorname{Log}^k = \operatorname{Log}^{k-1}$).

- *⌈calculer l'opérateur local et l'ordre de troncature des séries correspondant⌋*

1 se ramener à $z_0 = 0$: calculer $L(z, \theta) \in \mathbb{K}[z]\langle\theta\rangle$ annulant les $y(z_0 + z)$, $y \in \ker D$

2 en utilisant la série g, calculer N tel que la matrice M obtenue à l'issue de la ligne 22 vérifie $\|M - M(D, z_0 \to z_1)\|_{\mathrm{F}} \leqslant \varepsilon/2$ *⌈je ne détaille pas ce calcul pour éviter d'introduire encore plus de notations lourdes⌋*

- *⌈parcourir les solutions canoniques⌋*

3 calculer la décomposition sans décalage (8.25) de $Q_0 = [S_n^0]\, L(S_n^{-1}, n)$ (on convient que $e_{f,\sigma} = 0$ pour $\sigma \notin \Sigma_f$)

4 pour $f \in F$ et $g \in \mathbb{K}[n]$ facteur irréductible de f

- *⌈préparer la matrice de la récurrence dans le cas générique⌋*

5 $t := \sum_{\sigma \in \Sigma_f} e_{f,\sigma}$; et soit Log l'indéterminée de $\mathbb{K}[[\operatorname{Log}]]/\langle \operatorname{Log}^t \rangle$

6 $\lambda^{\mathrm{den}} := \operatorname{lc}(g)$; $\hat{g} := \operatorname{Numer}\big(g\big(n/(\operatorname{lc} g)\big)\big)$ *⌈ainsi, le polynôme $\hat{g}$ est unitaire et annule $\lambda^{\mathrm{den}}\,\lambda$ pour tout λ tel que $g(\lambda) = 0$⌋* ; et soit λ^{num} une variable symbolique représentant un générateur de $\mathbb{K}[\lambda^{\mathrm{num}}]/\langle \hat{g}(\lambda^{\mathrm{num}}) \rangle$

7 calculer en fonction de $\nu = n - \lambda^{\mathrm{num}}/\lambda^{\mathrm{den}}$ la matrice $B(n)$ pour $\mu(n) = 0$

8 initialiser une table d'association U *⌈qui va stocker les sommes partielles « abstraites », c.-à-d. dépendant d'algébriques non plongés dans $\mathbb{C}$⌋*

9 faire $U_{[0,v]} \leftarrow {}^{\mathrm{t}}(0, ..., 0, \operatorname{Log}^v, 0)$ pour $0 \leqslant v < e_{f,0}$

10 pour chaque paire $(\sigma_{\mathrm{pre}}, \sigma_{\mathrm{cur}})$ d'éléments consécutifs de $\Sigma_f \cup \{N\}$, par ordre croissant

- *⌈dérouler la récurrence, traiter l'indice exceptionnel $\lambda + \sigma_{\mathrm{cur}}$⌋*

11 calculer $P(\sigma_{\mathrm{pre}} + 1, \sigma_{\mathrm{cur}})$ par scindage binaire

12 $P(\sigma_{\mathrm{pre}} + 1, \sigma_{\mathrm{cur}} + 1) := B^*(\lambda^{\mathrm{num}}/\lambda^{\mathrm{den}} + \sigma_{\mathrm{cur}})\, P(\sigma_{\mathrm{pre}} + 1, \sigma_{\mathrm{cur}})$

- *⌈prolonger les solutions connues⌋*

13 pour $[\sigma, v] \in \operatorname{Indices}(U)$

14 $U_{[\sigma,v]} \leftarrow P(\sigma_{\mathrm{pre}} + 1, \sigma_{\mathrm{cur}} + 1) \cdot U_{[\sigma,v]}$

15 décaler l'avant-dernière ligne de $U_{[\sigma,v]}$ par $S_k^{-e_{f,\sigma_{\mathrm{cur}}}}$, en complétant par des zéros

- *⌈ajouter les solutions de valuation $\lambda + \sigma_{\mathrm{cur}}$⌋*

16 faire $U_{[\sigma_{\mathrm{cur}},v]} \leftarrow {}^{\mathrm{t}}(0, ..., 0, \operatorname{Log}^v, 0)$ pour $0 \leqslant v < e_{f,\sigma_{\mathrm{cur}}}$

17 faire $U_{[\sigma,v]} \leftarrow {}^{\mathrm{t}}(0, ..., 0)$ pour $\sigma \in \Sigma_f$ et $0 \leqslant v < e_{f,\sigma}$ tels que $\sigma > N$

- *⌈remplacer les algébriques abstraits par leurs valeurs complexes et développer les facteurs $\log(z_1 + \delta)$⌋*

18 pour $[\sigma, v] \in \operatorname{indices}(U)$ et ζ tel que $\hat{g}(\zeta) = 0$

19 soit u la dernière entrée du vecteur $U_{[\sigma,v]}$ où Log^k est remplacé par $\frac{1}{k!} \log^k(z_1 + \delta)$ pour tout $k \in [\![0, t-1]\!]$

20 calculer le développement en série tronqué $\tilde{u} \in \mathbb{K}[\lambda^{\mathrm{num}}][[\delta]]/\langle \delta^t \rangle$ du produit $(z_1 + \delta)^{\lambda+\sigma}\, u$ où $\lambda = \lambda^{\mathrm{num}}/\lambda^{\mathrm{den}}$ *⌈cela induit des calculs dans $\mathbb{K}(\lambda)$, d'où l'importance de le faire avant de spécialiser λ⌋*

21 $Y_{[\lambda+\sigma,v]} := ([\delta^j]\,\tilde{u})_{0 \leqslant j < r}$ où ζ est substitué à λ^{num}

- *⌈déduire des sommes partielles obtenues la matrice de transition⌋*

22 former la matrice M de colonnes $Y_{[n,v]}$, triées par indice dans l'ordre (8.18) *⌈M contient à ce stade des algébriques et des éléments $\log z_1$ non évalués⌋*

23 calculer une valeur approchée $\tilde{M} \in \mathbb{K}^{r \times r}$ de M telle que $\|\tilde{M} - M\|_{\mathrm{F}} \leqslant \varepsilon/2$

24 renvoyer $\tilde{M}$ ♢

à multiplier présentée en §8.4.2 comme suit. Ci-dessus, on a noté $K = (0, ..., 0, 1)$. Pour tout $n \in \mathbb{C}$ fixé, la matrice $C(n)$ est définie par

$$C(n) = \begin{pmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ -TQ_s(n+S_k) & -TQ_{s-1}(n+S_k) & \dots & -TQ_1(n+S_k) \end{pmatrix} \in \mathbb{K}[n][[S_k]]/\langle S_k^t \rangle$$

où l'on a posé (cf. §4.4)

$$L(S_n^{-1}, n) = Q_0(n) + Q_1(n)\, S_n^{-1} + \cdots + Q_s(n)\, S_n^{-s}$$

$$\mu(n) = \nu(n, Q_0) \in \mathbb{N} \qquad t = \sum_{\sigma \in \mathbb{Z}} \mu(n+\sigma) \in \mathbb{N}$$

$$T = S_k^{\mu(n)}\, Q_0(n+S_k)^{-1} \bmod S_k^t \in \mathbb{K}[n][[S_k]]/\langle S_k^t \rangle.$$

Si les $y_{n,k}$ sont les coefficients d'une solution quelconque de $D \cdot y = 0$, et en notant y_n la suite $(y_{n,k})_{k\in\mathbb{N}}$, on a

$$\begin{pmatrix} y_{n-s+1} \\ \vdots \\ y_{n-1} \\ S_k^{\mu(n)} \cdot y_n \end{pmatrix} = C(n) \begin{pmatrix} y_{n-s} \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix}$$

d'après la proposition 4.16. De même, l'application de $B(n)$ au vecteur

$$^{\mathrm{t}}(z^n\, y_{n-s}, ..., z^n\, y_{n-2}, z^n\, y_{n-1}, \Sigma_{n-1}) \tag{8.23}$$

où Σ_n est la suite des coefficients (qui sont polynomiaux en z) de $\log^k z/k!$ dans la somme partielle $y_{;n-1}(z)$ donne $^{\mathrm{t}}(z^n\, y_{n-s+1}, ..., z^n\, y_{n-1}, z^n\, S_k \cdot y_{n-1}, \sigma_n)$.

Il est clair que la forme de $B(n)$ ne dépend que de $\mu(n)$ et de t. Notamment, à $n \bmod 1$ fixé, on peut calculer en fonction d'un paramètre formel n une matrice $B(n)$ valable pour tout n tel que $\mu(n) = 0$. C'est ce que font les lignes 5 et 7 de l'algorithme 8.23. Restent les indices exceptionnels, en nombre fini, où $\mu(n) > 0$. On calcule séparément les matrices $B(n)$ correspondantes (ligne 12), notées aussi $B^*(n)$ pour les mettre en évidence.

Indices algébriques. Alors que les indices n où étaient évaluées les matrices $B(n)$ dans le cas d'un point ordinaire étaient des entiers, pour un point singulier régulier, il s'agit en général de nombres algébriques. Cela ne se ressent guère dans la description mathématique, où l'on se contente de sommer sur $n \in \lambda + \mathbb{Z}$ au lieu de $n \in \mathbb{N}$. Pour le calcul, cependant :

(i) on se ramène à des indices entiers en posant $n = \lambda + \nu$ avec $\nu \in \mathbb{N}$;

(ii) on sépare numérateur et dénominateur, ce qui conduit à représenter λ non pas par son polynôme minimal mais comme le quotient d'un entier algébrique λ^{num}, lui-même donné par son polynôme minimal, et d'un dénominateur λ^{den} (voir l'étape 6 de l'algorithme).

On prend pour λ l'élément de partie réelle minimale d'une classe de congruence modulo 1 de racines de Q_0. Les racines de la même classe sont notées $\lambda + \sigma$, $\sigma \in \mathbb{N}$.

Codage et multiplication des matrices. Au final, comme dans l'algorithme 8.12, on représente les matrices $B(n)$ et leurs produits par des quintuplets $(\Delta, d, q, p, \Lambda)$. En ce qui concerne $B(n)$, on a

$$\begin{cases} \Delta = \mathrm{Numer}(C) & \in (\mathbb{A}[\lambda^{\mathrm{num}}, \nu][[S_k]]/\langle S_k^t \rangle)^{s\times s} \\ d = \mathrm{Denom}(C) & \in \mathbb{A}[\lambda^{\mathrm{num}}, \nu] \\ q = \mathrm{Denom}(z_1 - z_0) & \in \mathbb{Z} \\ p = \mathrm{Numer}(z_1 - z_0) + q\,\delta & \in \mathbb{A}[[\delta]]/\langle \delta^r \rangle \\ \Lambda = (0, 0, ..., d\,q) & \in (\mathbb{A}[\lambda^{\mathrm{num}}, \nu][[S_k, \delta]]/\langle S_k^t, \delta^r \rangle)^{1\times s}. \end{cases} \tag{8.24}$$

Observons que le calcul de cette représentation à partir de la définition (8.22) de $B(n)$ fait appel à quelques manipulations de nombres algébriques, et conduit notamment à des normalisations d'éléments de corps de nombres.

La formule de multiplication des quintuplets $(\Delta, d, q, p, \Lambda)$ qui traduit le produit de matrices est la même que dans l'algorithme 8.12. Elle préserve les domaines de chacun des éléments indiqués dans l'équation (8.24).

En comparaison de la version de van der Hoeven [236], notre représentation utilise des matrices de coefficients $C(n)$ de côté divisé par t, en échange de l'introduction de coefficients séries tronquées. La complexité du produit, et donc (sous des hypothèses convenables) celle de l'algorithme, en diminue d'un facteur $t^{\omega-1}$. Cette amélioration se combine aux avantages de l'écriture (8.24) déjà mentionnés dans le cas d'une série quelconque ou d'un point ordinaire.

Remarque 8.24. J'ai conservé dans cette section $z_0 \in \mathbb{K}$ par cohérence avec le reste du manuscrit et l'état actuel de l'implémentation dans NumGfun. Cependant, pour traiter en toute généralité les points singuliers réguliers d'équations à coefficients rationnels, il faudrait permettre à z_0 lui-même d'être algébrique. Cela ne pose pas de difficulté de principe [236], puisque le scindage binaire fonctionne sur un corps de nombres. On remplace alors $\mathbb{K}$ par $\mathbb{Q}(i, z_0)$ ci-dessus. ♢

Parcours des solutions canoniques. Voyons maintenant comment construire les sommes partielles des solutions canoniques en formant des produits de matrices $B(n)$ et en les appliquant à des vecteurs de conditions initiales convenables. En un point ordinaire, les différentes solutions canoniques correspondaient simplement à des choix différents de $u_{-s'}, ..., u_r$ comme conditions initiales de la récurrence. Il suffisait donc pour les obtenir toutes de calculer une fois pour toutes le produit $B(N-1)\cdots B(0)$.

En un point singulier régulier, les différences de valuation entre éléments de la base canonique peuvent être plus grandes que l'ordre de la récurrence, ou non entières. Afin de partager tout de même ce qui peut l'être entre le calcul des solutions canoniques, on est conduit à regrouper les exposants possibles suivant deux critères.

(i) Par classe d'équivalence modulo 1. Les solutions canoniques qui correspondent à des racines de Q_0 différant d'un entier s'obtiennent en déroulant une seule fois une même récurrence et en appliquant le résultat à des conditions initiales différentes (étapes 10 à 17). La nouveauté est que ces conditions « initiales » s'appliquent à des rangs donnés par les racines de Q_0, qui peuvent être arbitrairement espacés.

(ii) Par classe de conjugaison au sens de Galois. Toute la partie purement algébrique du calcul, qui ne distingue pas entre les exposants racines d'un même facteur irréductible de l'équation indicielle (étapes 5 à 17), est partagée entre ces exposants. Cela compense en première approximation le fait que les opérations dans un corps de nombres de degré d sont $\Theta(d)$ fois plus coûteuses que celles sur les rationnels. Ce n'est qu'à la fin (étapes 18 à 21) que chaque série solution exprimée en fonction d'un algébrique abstrait est copiée et spécialisée pour chaque racine complexe du polynôme minimal.

Moyennant ce partage, le coût du calcul de la matrice de transition est plus élevé qu'en un point ordinaire d'un facteur de l'ordre du nombre de racines de Q_0 qui ne s'obtiennent pas les unes à partir des autres par décalage entier.

On organise le parcours des solutions canoniques en s'appuyant sur une factorisation de l'équation indicielle appelée décomposition sans décalage.

Proposition 8.25. [108] Un polynôme $Q_0 \in \mathbb{K}[n]$ admet une unique *décomposition sans décalage* (*shiftless decomposition*)

$$Q_0(n) = c \prod_{f \in F} \prod_{\sigma \in \Sigma_f} f(n+\sigma)^{e_{f,\sigma}} \tag{8.25}$$

où $c \in \mathbb{K}$, $F \subset \mathbb{K}[n]$, $0 \in \Sigma_f \subset \mathbb{N}$, $e_{f,\sigma} \in \mathbb{N}^*$ et

- les polynômes $f \in F$ sont unitaires et sans facteur carré ;
- lorsque $f, g \in F$ et $f \neq g$, les polynômes $f(n)$ et $g(n+\sigma)$ sont premiers entre eux pour tout $\sigma \in \mathbb{Z}$. ♢

Indices exceptionnels et conditions initiales. On ne peut se contenter de former le produit $B(N-1)\cdots B(1)\,B(0)$ pour accumuler les termes des sommes partielles à calculer. En effet, lorsque n est un indice exceptionnel, l'application de $B^*(n)$ à un vecteur de la forme (8.23) donne en avant-dernière position $S_k^{\mu(n)} \cdot y_n$ et non y_n. Avant de repartir du vecteur obtenu pour calculer des termes plus lointains, on corrige ce décalage à la ligne 15 de l'algorithme. Pour une solution y arbitraire, les $\mu(n)-1$ termes à ajouter en tête de la suite y_n sont donnés par les conditions initiales généralisées. Dans l'algorithme 8.23, toutes les solutions considérées sont des éléments de la base canonique, pour lesquels une et une seule des conditions initiales généralisées est non nulle. On prolonge ainsi les solutions « déjà introduites », pour lesquelles les termes en question sont nuls, puis (ligne 16) on ajoute celles de valuation n, dans lesquelles l'un d'entre eux est égal à 1.

Facteurs irrationnels. Dans le cas d'un point ordinaire, les sommes partielles (exactes) de séries que nous obtenions comme approximations des entrées de chacune des matrices de transition étaient des éléments de $\mathbb{K}$. Ici, la présence d'algébriques et de termes dépendant du logarithme de z_1 font que ce n'est plus le cas. On se donne un peu de marge dans la précision à laquelle on calcule les sommes partielles pour pouvoir les approcher à leur tour par des complexes à coordonnées rationnelles. Cette dernière approximation fait l'objet de l'étape 23. Une façon simple de l'implémenter est d'évaluer numériquement l'expression en arithmétique d'intervalles, en doublant la précision du calcul jusqu'à arriver à la borne d'erreur voulue.

REMARQUE 8.26. Essentiellement le même algorithme permet de calculer les développements en séries logarithmiques des solutions. Il suffit de remplacer le point $z_1 \in \mathbb{K}$ par une indéterminée, et d'ignorer les étapes de contrôle de la précision. Quand on ne s'intéresse qu'aux développements formels, il est plus simple de calculer « directement » les $y_{n,k}$ par récurrence (voir §4.4.2). En revanche, dans une implémentation qui fournit à la fois ceux-ci et l'évaluation numérique, réutiliser l'approche par scindage binaire pour les développements formels peut faciliter le partage de code sans changer notablement le coût. C'est ainsi que fonctionne la procédure `local_basis` de NumGfun illustrée en section 1.2.6. ♢

8.6.3 Matrice de transition d'un point ordinaire vers un point singulier

Il y a une petite difficulté supplémentaire dans le cas du passage d'un point ordinaire z_1 à un point singulier z_0. Faute d'expression explicite analogue à (8.20) pour les coefficients de $M(D, z_1 \to z_0)$, on la calcule comme $M(D, z_0 \to z_1)^{-1}$, ce qui nécessite de relier erreur sur $M(D, z_0 \to z_1)$ et erreur sur son inverse.

La correction de l'algorithme 8.27 repose sur le lemme suivant.

LEMME 8.28. Soient A une matrice inversible et $\tilde{A}$ telle que $\|\tilde{A} - A\|_{\mathrm{F}} \leqslant \eta$. Si

(a) $\tilde{A}$ est inversible ;

(b) $\|\tilde{A}^{-1}\|_{\mathrm{F}} < \eta^{-1}$;

(c) $\|\tilde{A}^{-1}\|_{\mathrm{F}}\, \eta\big(\|\tilde{A}^{-1}\|_{\mathrm{F}} + \varepsilon\big) < \varepsilon$

alors on a $\|A^{-1} - \tilde{A}^{-1}\|_{\mathrm{F}} \leqslant \varepsilon$. ♢

DÉMONSTRATION. Posons $A = (1-H)\,\tilde{A}$. On a

$$\|H\|_{\mathrm{F}} = \|\big(\tilde{A} - A\big)\,\tilde{A}^{-1}\|_{\mathrm{F}} \leqslant \eta\,\|\tilde{A}^{-1}\|_{\mathrm{F}} < 1.$$

Algorithme 8.27. InverseSingularStepTransitionMatrix$(L, z_0, z_1, \varepsilon)$

Entrée. Un opérateur $D \in \mathbb{Q}[z]\langle\partial\rangle$, un point *régulier* éventuellement singulier $z_0 \in \mathbb{K}$ et un point *ordinaire* $z_1 \in \mathbb{K}$ de D, une borne d'erreur ε.

Sortie. Une approximation $\tilde{M}$ de la matrice de transition $M = M(D, z_1 \to z_0)$ à précision $\|\tilde{M} - M\|_{\mathrm{F}} \leqslant \varepsilon$.

1 $\lambda \leftarrow \frac{1}{10}$; $\varepsilon' \leftarrow \lambda\,\varepsilon$ *[ou mieux : prendre une estimation* $\lambda \approx \|M\|_{\mathrm{F}}^{-2}$ *telle que* $\lambda < 1$*]*

2 $\tilde{M}_{z_0 \to z_1} :=$ SingularStepTransitionMatrix$(L, z_0, z_1, \varepsilon')$

3 essayer

  4 calculer une approximation $\tilde{M}_{z_1 \to z_0}$ de $\tilde{M}_{z_0 \to z_1}^{-1}$ à précision $\|\tilde{M}_{z_1 \to z_0} - \tilde{M}_{z_0 \to z_1}^{-1}\|_{\mathrm{F}} \leqslant \frac{1}{2}\,\varepsilon$

5 en cas d'échec ($\tilde{M}_{z_0 \to z_1}$ n'est pas inversible)

  6 $\lambda \leftarrow \lambda^2$; $\varepsilon' \leftarrow \lambda\,\varepsilon'$

  7 revenir à l'étape 2

8 calculer $B \geqslant \|\tilde{M}_{z_1 \to z_0}\|_{\mathrm{F}}$

9 si $B \geqslant (\varepsilon')^{-1}$ ou $B\,(B + \varepsilon)\,\varepsilon' \geqslant \varepsilon$

  10 $\varepsilon' \leftarrow \min\,(B^{-2}, \lambda)\,\varepsilon'$

  11 revenir à l'étape 2

12 renvoyer $\tilde{M}_{z_1 \to z_0}$ ♢

En développant la différence sous la forme

$$\tilde{A}^{-1} - A^{-1} = \tilde{A}^{-1}\left(1 - (1 - H)^{-1}\right) = -\tilde{A}^{-1}\sum_{n=1}^{\infty} H^n$$

il vient

$$\|\tilde{A}^{-1} - A^{-1}\|_{\mathrm{F}} \leqslant \|\tilde{A}^{-1}\|_{\mathrm{F}}\,\frac{\|H\|_{\mathrm{F}}}{1 - \|H\|_{\mathrm{F}}} \leqslant \frac{\eta\,\|\tilde{A}^{-1}\|_{\mathrm{F}}^2}{1 - \eta\,\|\tilde{A}^{-1}\|_{\mathrm{F}}} \leqslant \frac{\|\tilde{A}^{-1}\|_{\mathrm{F}}\,\dfrac{\varepsilon}{\|\tilde{A}^{-1}\|_{\mathrm{F}} + \varepsilon}}{1 - \dfrac{\varepsilon}{\|\tilde{A}^{-1}\|_{\mathrm{F}} + \varepsilon}} = \varepsilon$$

d'après l'hypothèse (c). □

Proposition 8.29. Les algorithmes 8.23 et 8.27 s'exécutent en $O(M(p \lg^2 p))$ opérations, où $p = \lg\,(\varepsilon^{-1})$. ♢

Démonstration (esquisse). Dans l'algorithme 8.23, on a $N = O(\varepsilon^{-1})$ d'après la proposition 6.20 page 118. Les seules étapes coûteuses vis-à-vis de ε sont les lignes 11 et 23. La première prend $O(M(N \lg^2 N))$ opérations. Lors de la seconde, la précision des calculs flottants intermédiaires est $\Omega(\varepsilon)$; or, le calcul à 2^{-n} près d'une racine d'un polynôme fixé prend $O(M(n))$ opérations par la méthode de Newton, et le logarithme se calcule à précision 2^{-n} en $O(M(n) \lg n)$ opérations [40]. Chacune de ces lignes est parcourue $O(1)$ fois. Dans l'algorithme 8.27, le nombre d'itérations ne dépend pas de ε, et l'on a toujours $\varepsilon' = \Omega(\varepsilon)$. Le coût de l'inversion de matrice de l'étape 4 est de $O(M(p) \lg p)$ opérations binaires d'après le théorème 7.7. □

8.7 Applications

8.7.1 Évaluation de fonctions D-finies

Dans l'optique de ce mémoire, l'application phare du prolongement analytique numérique est évidemment l'évaluation des fonctions D-finies. Nous avons vu de nombreux exemples d'évaluation à grande précision dans le chapitre 1 et dans celui-ci.

Revenons cependant un instant sur les évaluations répétées à précision modérée. L'idée ici est de tirer profit du prolongement analytique pour calculer une approximation polynomiale *garantie* d'une fonction donnée en un point éloigné de celui où sont données les conditions initiales qui la définissent. La fonction `diffeqtoproc` présentée en §1.2.5 offre une implémentation assez naïve de ce schéma. Elle prend en entrée une fonction D-finie y, une précision de précalcul ε, et une liste de disques $D_k \subset \mathbb{C}$. Le disque $D_k = \{z : |z - c_k| < \rho_k\}$ est donné par un chemin de l'origine à son centre c_k et un rayon ρ_k. Les disques ne doivent pas contenir de points singuliers de l'équation. Pour chaque disque D_k, `diffeqtoproc` calcule un polynôme $p_k \in \mathbb{K}[z]$ tel que

$$\forall z \in D \cap \mathbb{K}, \quad |\diamond(p_k(z)) - y(z)| \leqslant \varepsilon$$

où $\diamond$ désigne l'arrondi au complexe à coordonnées décimales le plus proche. La procédure renvoyée prend en entrée un point $z \in \mathbb{K}$ et une précision cible ε. Elle teste si $\varepsilon \leqslant \varepsilon'$ et successivement si $z \in D_1, D_2, \ldots$, renvoie une approximation décimale de $p_k(z)$ le cas échéant, et appelle `analytic_continuation` sinon.

EXEMPLE 8.30. Le tracé en figure 8.3 de la fonction y définie par l'équation

$$(z-1)\,y'''(z) - 2\,(2\,z-5)\,y''(z) - (4\,z-6)\,y'(z) + z^2\,(z-1)\,y(z) = 0$$

et les conditions initiales

$$y(0) = 2, \quad y'(0) = 1, \quad y''(0) = 0$$

a été obtenu en appelant `diffeqtoproc` avec une liste de disques (choisie à la main) qui recouvrent le domaine du graphique en évitant le point $z = 1$.

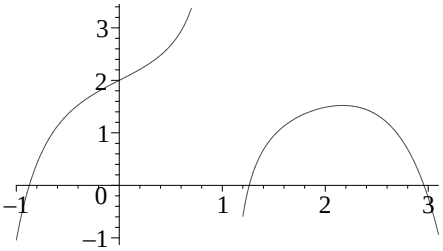


Figure 8.3. Intégration d'une équation différentielle au-delà d'un pôle.

La disponibilité du prolongement analytique numérique nous permet de contourner le pôle en $z = 1$. Un intégrateur numérique simple serait au contraire bloqué par le pôle et ne parviendrait pas à évaluer la fonction à droite de $z = 1$. ♢

Dans l'état actuel des choses, les polynômes p_k sont calculés comme des combinaisons linéaires de séries de Taylor tronquées des solutions de base $y[c_k, j]$, avec des coefficients obtenus par prolongement analytique le long du chemin de 0 à c_k. Le contrôle des erreurs est similaire à celui de la section 8.3. On commence par calculer une borne B_{can} sur les valeurs sur D_k des solutions de base et de leurs r premières dérivées, où r est l'ordre de l'équation qui définit y. On choisit $\varepsilon' \leqslant \varepsilon/(40\ r)$. Le vecteur $Y(c_k)$ des « conditions initiales » en c_k est calculé à précision $\varepsilon'/B_{\text{can}}$, et l'on majore sa norme par $B_{\text{ini}} \geqslant \|Y(c_k)\|_{\text{F}}$. Enfin, on développe les $y[c_k, j]$ à un ordre N tel que $|y[c_k, j]_{N;}(z)| \leqslant \varepsilon'/B_{\text{ini}}$ pour tout $z \in D_k$, et l'on forme la combinaison linéaire des ε' dont les coefficients sont donnés par $Y(c_k)$. Tous ces calculs sont effectués dans $\mathbb{K}$, sans arrondis. On a donc $|p_k(z) - y(z)| \leqslant \varepsilon/20$ pour $z \in D$.

À ce stade, cependant, le polynôme p_k a généralement un degré plus grand que nécessaire. Aux précisions où il est rentable de précalculer une approximation polynomiale, nos bornes peuvent en effet être fort pessimistes. Si $p_k = \sum_{i=0}^{d} c_i\, z^i$, on a habituellement $|c_i\, \rho^i| \ll \varepsilon$ pour $i \approx d$. Afin d'améliorer la compacité et l'efficacité de

la procédure renvoyée, on supprime les coefficients $c_d, c_{d-1}, \ldots$ tant que

$$|c_d|\,\rho_k^d + |c_{d-1}|\,\rho_k^{d-1} + \cdots < \varepsilon/20.$$

Le polynôme tronqué (encore noté p_k) obtenu satisfait $|p_k(z) - y(z)| \leqslant \varepsilon/10$ pour tout $z \in D$, ce qui laisse la marge nécessaire à l'arrondi décimal final (lemme 8.8).

Les opportunités d'améliorations ne manquent pas. La hauteur du polynôme p_k construit par le procédé ci-dessus est elle aussi plus grande que l'ordre de grandeur que l'on attendrait, soit $\log(\varepsilon^{-1})$. Il serait naturel de remplacer les « gros » coefficients par des approximations à précision suffisante. Plus généralement, au lieu de seulement jeter les coefficients de degré élevé superflus, on pourrait approcher à son tour p_k par un polynôme $\tilde{p}$ aussi compact que possible — fût-ce de manière heuristique, dans la mesure où il n'est pas difficile de borner *a posteriori* la différence $p_k - \tilde{p}$.

Par ailleurs, partir du développement de Taylor tronqué de y est un choix plus ou moins raisonnable suivant le type d'approximation recherché. Les séries de Taylor tronquées ne sont pas loin de minimiser l'erreur en norme uniforme sur les disques, à degré donné [107]. Mais de nombreuses applications requièrent des évaluations répétées sur d'autres sortes de domaines, et notamment, comme dans l'exemple ci-dessus, sur des segments. Nous étudierons au chapitre suivant une technique de calcul d'approximations polynomiales de fonctions D-finies sur des segments efficace jusqu'à des degrés élevés. Celle-ci pourrait être utilisée dans `diffeqtoproc` à l'avenir, de même que des variantes pour d'autres sortes de domaines[5].

Plus simplement, on peut appliquer un analogue de l'élimination des termes superflus décrite ci-dessus [154, §VII.10], connu sous le nom d'*économisation*. L'idée est de faire baisser le degré du polynôme en lui soustrayant, non pas des monômes, mais des *polynômes de Tchebycheff* adaptés au segment I sur lesquel on travaille, qui sont de norme uniforme minimale. On arrive à des approximations valables seulement sur I, mais plus compactes que celles données directement par les séries de Taylor tronquées. Reste que cette idée ne fonctionne, en l'état, que pour un segment entièrement contenu dans le disque de convergence du développement en série entière initial.

8.7.2 Monodromie

En sortant un peu du cadre des *fonctions* D-finies, le prolongement analytique à grande précision sert à étudier les équations différentielles. Il s'agit d'ailleurs historiquement d'une motivation majeure de son introduction en calcul formel [54, §7].

Les matrices de transition le long de boucles qui font le tour d'un point singulier exactement une fois, par exemple

```
> diffeq := (1+z^2)*diff(y(z),z,z)+2*z*diff(y(z),z)=0;   # (1, arctan)
```

$$(1+z^2)\left(\frac{\mathrm{d}^2}{\mathrm{d}z^2}y(z)\right)+2\,z\left(\frac{\mathrm{d}}{\mathrm{d}z}y(z)\right)=0$$

```
> transition_matrix(diffeq, y(z), [0,I+1,2*I,I-1,0], 30);
```

$$\begin{pmatrix} 1.00000000000000000000000000000000 & 3.14159265358979323846264338328 0 \\ 0. & 1.00000000000000000000000000000000 \end{pmatrix}$$

s'appellent matrices de monodromie locale. Mises ensemble, les matrices de monodromie locale autour de chacun des points singuliers basées en un même point engendrent le *groupe de monodromie* [245], qui intervient dans l'étude globale des équations différentielles linéaires. Leur calcul numérique à précision arbitraire est la brique de base d'une approche symbolique-numérique des équations différentielles linéaires pouvant aller jusqu'au calcul du groupe de Galois différentiel [238].

5. La méthode de validation du résultat utilisée pourrait s'avérer plus intéressante que nos bornes *a priori* même pour les disques. Voir aussi [240].

La détermination numérique de la monodromie locale par prolongement analytique autour de la singularité fonctionne même pour un point singulier irrégulier. Dans le cas régulier, on dispose d'une méthode alternative[6] qui utilise une matrice de transition du point singulier régulier vers un point ordinaire voisin [236, §4.2.5].

Reprenons l'exemple ci-dessus, et écrivons une matrice fondamentale Y de l'équation au voisinage du point singulier $z = i$:

```
> basis := local_basis(diffeq, y(z), I, 3):
> Y := Matrix([basis, diff(basis, z)]);
```

$$\begin{pmatrix} \ln(z) + \frac{1}{2}\mathrm{i}\,z - \frac{1}{8}z^2 - \frac{1}{24}\mathrm{i}\,z^3 & 1 \\ \frac{1}{z} + \frac{1}{2}\mathrm{i} - \frac{1}{4}z - \frac{1}{8}\mathrm{i}\,z^2 & 0 \end{pmatrix}$$

Après un tour dans le sens direct autour du point singulier, cette matrice est transformée en $Y^\sharp(z) = Y(z)\,C$, où

```
> C := Matrix([[1, 0], [2*Pi*I,1]]);
```

$$\begin{pmatrix} 1 & 0 \\ 2\,\mathrm{i}\,\pi & 1 \end{pmatrix}$$

```
> Y.C;
```

$$\begin{pmatrix} \ln(z) + \frac{1}{2}\mathrm{i}\,z - \frac{1}{8}z^2 - \frac{1}{24}\mathrm{i}\,z^3 + 2\,\mathrm{i}\,\pi & 1 \\ \frac{1}{z} + \frac{1}{2}\mathrm{i} - \frac{1}{4}z - \frac{1}{8}\mathrm{i}\,z^2 & 0 \end{pmatrix}$$

La matrice C se calcule aisément à partir de l'équation indicielle en i, seuls les facteurs $\ln z$ ou z^α, $\alpha \notin \mathbb{Z}$, qui apparaissent dans Y étant modifiés de façon non triviale lors du passage à $Y^\sharp$. D'après la règle de composition (8.9), on en déduit la matrice de monodromie locale comme suit :

```
> M1 := transition_matrix(diffeq, y(z), [0,I], 30):
> M2 := transition_matrix(diffeq, y(z), [I,0], 30):
> evalf[31](M2.C.M1);
```

$$\begin{pmatrix} 1.000000000000000000000000000000 + 0.\,\mathrm{i} & 3.141592653589793238462643383280 + 0.\,\mathrm{i} \\ 0. + 0.\,\mathrm{i} & 1.000000000000000000000000000000 + 0.\,\mathrm{i} \end{pmatrix}$$

On voit au passage plus clairement comment est apparu le nombre π dans la matrice de monodromie locale.

L'implémentation dans NumGfun est, à ma connaissance, la première capable de mener ce genre de calculs, du moins de manière garantie et à précision arbitraire. Des implémentations de calcul de monodromie par prolongement analytique numérique existent en revanche dans le cas particulier des équations algébriques, où l'on peut reconstruire *exactement* le groupe de monodromie (voir Poteaux [204] et ses références). Il serait intéressant d'automatiser complètement le calcul de générateurs du groupe de monodromie, en adaptant notamment certaines idées utilisées dans le cas algébrique pour choisir des chemins de prolongement de faible coût.

8.7.3 Asymptotique des suites P-récursives

Des logiciels récents, écrits par Zeilberger [266] en Maple et par Kauers [142] en Mathematica, proposent de calculer automatiquement un *développement asymptotique* à l'infini d'une suite P-récursive (u_n). Dans les deux cas, la méthode implémentée est celle de Birkhoff-Trjitzinsky (voir §3.4.3).

Dans certains cas — lorsque les singularités de l'équation différentielle associée à la récurrence qui limitent le domaine de convergence de la série génératrice de (u_n)

6. Elle devrait en général être plus efficace d'un facteur constant (c.-à-d. indépendant de la précision), mais il est difficile de le confirmer dans l'état actuel de mes implémentations.

sont des points singuliers réguliers à distance finie de l'origine — on peut procéder à la place par *analyse de singularité* [137, 91, 94]. L'idée est de *transférer* aux coefficients de leurs développements en série le comportement asymptotique, bien compris (voir §3.4.1 et chapitre 5), des *fonctions D-finies* au voisinage de leurs singularités. Il est bien connu que cette méthode est elle aussi algorithmique [93, 94], même si elle n'a pas, à ma connaissance, été automatisée dans le cas des fonctions D-finies. Le programme gdev de Salvy [213, 205] l'implémente pour une large classe d'expressions explicites.

Le problème des constantes. Que l'on passe par l'une ou l'autre de ces approches, ce que l'on obtient en fait à partir d'une récurrence est une *base* des comportements asymptotiques des solutions. La décomposition dans cette base de la solution particulière objet du calcul est plus délicate à déterminer. On ne sait pas le faire exactement, sauf cas exceptionnels.

Dans le cas le plus simple, la base trouvée comporte un unique développement asymptotique dominant. On parvient alors généralement à estimer la valeur numérique de la constante manquante en comparant les valeurs prises par la suite pour des indices assez grands aux quelques premiers termes de ce développement [142, §3]. La situation se complique s'il faut combiner plusieurs contributions du même ordre de grandeur — disons, si $u_n \sim (\alpha + (-1)^n \beta)\, 2^n$. De même lorsque le plus divergent des comportements asymptotiques de la base ne correspond en fait pas à celui de la suite u_n, c'est-à-dire, lorsque le coefficient correspondant de la combinaison linéaire est nul. Des interventions manuelles sont nécessaires pour exhiber un développement vraisemblable. En tout état de cause, dans les implémentations existantes, le calcul des coefficients de la décomposition est heuristique.

Flajolet et Puech [92, §5.4] décrivent cependant le principe d'une méthode pour le faire, toujours numériquement, mais de façon rigoureuse. Elle s'applique à l'approche par analyse de singularité. L'idée consiste à déterminer par connexion numérique les coefficients de l'expression d'une fonction D-finie au voisinage de points singuliers convenables (supposés réguliers) de l'équation différentielle associée. On déduit de cette expression le comportement asymptotique précis de la fonction à chacune des singularités concernés, puis celui des coefficients de son développement à l'origine. La présentation de Flajolet et Puech laisse cependant ouverts nombre de détails. L'objet de cette section est de montrer, sur un exemple, comment mettre en œuvre leur stratégie à l'aide des outils développés dans ce chapitre. On peut ainsi calculer efficacement, à grande précision et de façon garantie, des approximations numériques des constantes en tête des développements asymptotiques.

Le problème analogue dans le cas des fonctions algébriques (pour lequel la connexion aux singularités est faisable de façon exacte) est étudié dans la troisième partie de la thèse de Chabaud [46]. Banderier *et al.* [11] ont aussi annoncé un travail en cours sur le problème des constantes, par des méthodes d'accélération de convergence.

Analyse de singularité. Le cœur de la méthode d'analyse de singularité que nous nous proposons d'appliquer est le théorème de transfert suivant. On peut le voir comme un raffinement de la règle de Cauchy (proposition 3.22) suivant laquelle le coefficients u_n d'une série de rayon de convergence égal à 1 doivent satisfaire $u_n = e^{o(n)}$ quand $n \to \infty$.

Théorème 8.31. [94, Theorem VI.3] Soit

$$\Delta = \{z : (|z| < R) \wedge (z \neq 1) \wedge (|\arg(z-1)| > \varphi)\} \quad \text{où} \quad R > 1, \varphi \in]0, \frac{\pi}{2}[$$

le domaine du plan complexe représenté en figure 8.4. Soit ε une fonction analytique sur Δ telle que

$$\varepsilon(z) = O\left((1-z)^{-a} \log^k \frac{1}{1-z} \right), \qquad a, k \in \mathbb{R}$$

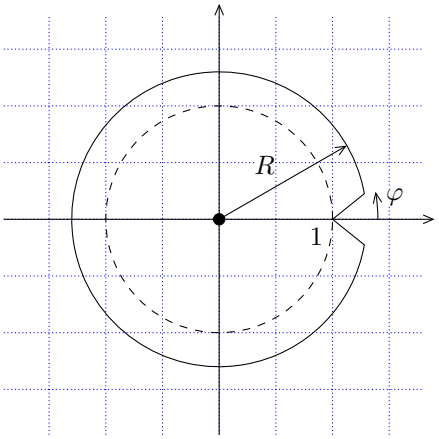


Figure 8.4. Le domaine Δ du théorème 8.31.

lorsque $z \to 1$ avec $z \in \Delta$. Alors les coefficients du développement en série à l'origine de ε satisfont

$$[z^n]\, \varepsilon(z) = O(n^{a-1} \log^k n)$$

quand $n \to \infty$. ♢

Considérons la fonction génératrice $u(z)$ d'une suite (u_n), et supposons $u(z)$ analytique sur Δ, avec une singularité en $z=1$. Si l'on a $u(z) = \sigma(z) + O(\varepsilon(z))$ quand $z \to 1$, où $\sigma = \sum_n \sigma_n z^n$ est une fonction « simple » elle-même analytique sur Δ, le théorème précédent permet de déduire un développement asymptotique de u_n quand $n \to \infty$ de celui de σ_n. Le résultat s'étend si u admet une unique singularité de module minimal, quitte à effectuer un changement de variable.

En présence d'un nombre fini de singularités de module minimal, les contributions de chacune s'additionnent, et on a le résultat suivant.

THÉORÈME 8.32. [94, Theorem VI.5] Soit u une fonction analytique à l'origine, avec un nombre fini de singularités $\zeta_1, \zeta_2, \ldots, \zeta_m$ de module ρ minimal, et qui admet un prolongement analytique à $U = \bigcap_i (\zeta_i \Delta)$, où Δ est défini par (8.31). Supposons que pour tout i, on a quand $z \to \zeta_i$ avec $z \in U$

$$u(z) = \sigma_i\left(\frac{z}{\zeta_i}\right) + O\left(\tau\left(\frac{z}{\zeta_i}\right)\right) \tag{8.26}$$

où chacune des fonctions σ_i et τ est combinaison linéaire de termes de la forme[7]

$$(1-z)^{-a}\left(\frac{1}{z}\log\frac{1}{1-z}\right)^k, \qquad a, k \in \mathbb{C}.$$

Alors, les coefficients de Taylor à l'origine de u satisfont

$$[z^n]\, u(z) = \zeta_1^{-n}\, [z^n]\, \sigma_1 + \cdots + \zeta_m^{-n}\, [z^n]\sigma_m + O(\rho^{-n}\, n^{a-1} \log^k n) \tag{8.27}$$

quand $n \to \infty$, où les paramètres a et k sont relatifs au terme d'erreur τ. ♢

Soulignons que la notation $[z^n]\ \sigma_i$ dans (8.27) fait référence aux coefficients de Taylor *à l'origine* des σ_i.

7. Le facteur $1/z$ devant $\log(1/(1-z))$ est là pour assurer l'analyticité à l'origine quand $k \notin \mathbb{Z}$: on a par exemple $\sqrt{\log(1/(1-z))} = z^{1/2} + z^{3/2}/4 + \cdots$, tandis que $\sqrt{\log(1/(1-z))/z} = 1 + z/4 + \cdots$. Il ne change rien à l'asymptotique de la fonction au voisinage de 1 ou à celle de la suite des coefficients. Voir Flajolet et Odlyzko [91, §3] pour plus de détails. Nous n'utilisons ici que le cas $k \in \mathbb{N}$.

D'après le théorème 4.11 (p. 63), toute série D-finie convergente $u(z)$ et solution d'une équation dont les singularités dominantes sont des points singuliers réguliers entre dans les conditions d'application de ce résultat. Dans le rôle des σ_i, on adopte les troncatures à un ordre convenable des séries logarithmiques calculées au chapitre 4. Le développement (8.26) est donc convergent dans le cas qui nous intéresse, mais cela n'a pas d'importance pour la suite. On dispose ainsi d'une méthode systématique pour accéder à l'asymptotique des coefficients $[z^n]\,u$. Dans le cas d'une série divergente ou d'une série entière, on peut tenter d'appliquer la normalisation présentée en §5.2.3 pour le calcul de bornes afin de se ramener à la situation précédente.

Asymptotique aux singularités algébrico-logarithmiques. Avant d'avoir tous les outils pour calculer explicitement les développements asymptotique promis, il reste à établir l'asymptotique des coefficients des fonctions σ_i que nous avons adoptées. La proposition suivante est due à Jungen [137] ; voir aussi Flajolet et Sedgewick [94, p. 387-389].

Proposition 8.33. [137] Pour $k \in \mathbb{N}^*$, les coefficients du développement en série à l'origine de la fonction $(1-z)^{-a}\log^k\left((1-z)^{-1}\right)$ admettent le développement asymptotique (au sens de la définition rappelée en §3.4.1) quand $N \to \infty$

$$
\begin{aligned}
&[z^N]\,(1-z)^{-a}\log^k\frac{1}{1-z}\\
&= \frac{\partial^k}{\partial a^k}\binom{N+a-1}{a-1}\\
&\sim \begin{cases} \frac{N^{a-1}}{\Gamma(a)}\left(E_0(\log N)+\frac{1}{N}E_1(\log N)+\cdots\right), & a\notin\mathbb{Z}_-\\ (-1)^a\,k\,\Gamma(1-a)\,N^{a-1}\left(\hat{E}_0(\log N)+\frac{1}{N}\hat{E}_1(\log N)+\cdots\right), & a\in\mathbb{Z}_- \end{cases}
\end{aligned}
$$

où les $E_i, \hat{E}_i$ sont des polynômes avec $\deg E_i \leqslant k$, $\deg \hat{E}_i \leqslant k-1$. ♢

Les polynômes E_i ou $\hat{E}_i$ sont calculables explicitement. Par exemple, le module Maple MultiSeries (successeur d'une partie du module gdev déjà cité) nous donne

```
> coef := binomial(N+a-1,a-1);
```

$$\binom{N+a-1}{a-1}$$

```
> simplify(MultiSeries:-series(subs(a=3/2, diff(coef, a, a)), N=infinity, 1), 'size');
```

$$\frac{2}{\sqrt{\pi}\left(\frac{1}{N}\right)^{\frac{1}{2}}}\left(4-\frac{\pi^2}{2}+\ln(N)^2-2\ln(N)\,(2-\gamma-2\ln(2))+(2-\gamma-2\ln(2))^2\right)+O\left(\frac{\ln(N)^2}{\sqrt{N}}\right)$$

pour $a=3/2$ et $k=2$.

Un exemple. L'analyse de singularité automatique, avec constantes, pour les fonctions D-finies est en cours d'implémentation dans NumGfun. Bornons-nous ici à reprendre « à la main » l'un des exemples proposés par Zeilberger dans la documentation de son programme AsyRec[8], à savoir la suite $(u_n)_{n\geqslant 1}$ définie par

```
> rec := {(n+2)^2*u(n+2)-(7*n^2+21*n+16)*u(n+1)-8*(n+1)^2*u(n), u(1)=2, u(2)=10};
```

$$\{(n+2)^2\,u(n+2)-(7\,n^2+21\,n+16)\,u(n+1)-8\,(n+1)^2\,u(n), u(1)=2, u(2)=10\}$$

dont voici les quelques premiers termes

```
> p := rectoproc(rec, u(n), 'remember'):
> seq(p(i), i=1..10);
```

8. Exécuter la commande `ezra(AsyC)`.

2,10,56,346,2252,15184,104960,739162,5280932,38165260

Ces nombres sont appelés nombres de Franel (OEIS A000172 [220]).

L'équation différentielle associée par la substitution $S \mapsto z^{-1}, n \mapsto \theta$ à la récurrence précédente est

```
> deq := prettify(NumGfun:-utilities:-purerectodiffeq(rec, u(n), y(z)));
```

$$-z\,(z+1)\,(8\,z-1)\left(\frac{\mathrm{d}^2}{\mathrm{d}z^2}y\,(z)\right)+(-24\,z^2-14\,z+1)\left(\frac{\mathrm{d}}{\mathrm{d}z}y\,(z)\right)-2\,y\,(z)\,(4\,z+1)=0$$

L'origine est un point singulier régulier, et la base canonique de solutions correspondante, à savoir

```
> local_basis(deq, y(z), 0, 5);
```

$$\left[\ln\,(z)+(3+2\ln\,(z))\,z+\left(\frac{33}{2}+10\ln\,(z)\right)z^2+(100+56\ln\,(z))\,z^3+\left(\frac{2561}{4}+346\ln\,(z)\right)z^4,\right.$$
$$\left.1+2\,z+10\,z^2+56\,z^3+346\,z^4\right]$$

est formée d'une solution non-analytique et d'une solution analytique. On voit que cette dernière n'est autre que la série génératrice de (u_n) prolongée par $u_0=1$.

Développement en la singularité dominante. Il y a une unique singularité dominante, en $z=1/8$. On cherche donc l'expression au voisinage de cette dernière de la solution $y(z)=1+2\,z+\cdots$ analytique à l'origine. La connexion numérique entre les deux points singuliers donne

```
> sol := subs(z=w, analytic_continuation(deq, y(z), [0, 1/8], 10, ord=3));
```

$$(-.3675525969\ \_C_1)\left(\ln\,(w)+\left(\frac{-32}{9}-\frac{8}{3}\ln\,(w)\right)w+\left(\frac{1664}{81}+\frac{320}{27}\ln\,(w)\right)w^2\right)$$
$$+(-2.4183991523\ \_C_0+(.432914606e-1+1.1547005384\,\mathrm{i})\ \_C_1)\left(1-\frac{8}{3}w+\frac{320}{27}w^2\right)$$

expression qui représente le développement asymptotique quand $z \to \frac{1}{8}$, écrit en fonction de $w=z-\frac{1}{8}$, d'une combinaison linéaire des solutions canoniques en zéro.

Détail de quelque importance, d'après nos conventions sur la détermination du logarithme, chaque terme est analytique pour $w \in \mathbb{C} \setminus \mathbb{R}_-$. Le développement se rapporte au prolongement analytique de y « par au-dessus » de $z=\frac{1}{8}$. Or, nous avons besoin d'un développement analytique pour tout w de module assez petit et d'argument $|\arg w|>\varphi$, où $|\varphi|<\pi/2$, afin de pouvoir appliquer le théorème 8.31. On s'écarte donc exceptionnellement des conventions de la section 8.6.1, et l'on interprète ln dans la formule ci-dessus comme la détermination du logarithme analytique sur $\mathbb{C}\setminus\mathbb{R}_+$ qui coïncide avec le logarithme standard sur $\mathbb{R}_-$, c'est-à-dire

$$\ell(w)=\ln\,(-w)+\ln\,(-1)=\ln\,(-w)+i\,\pi.$$

Le remplacement est légitime puisque le chemin de prolongement suivi se trouve dans le domaine où $\ln w$ et $\ell(w)$ coïncident.

Effectuons le remplacement de $\ln w$ par $\ell(w)$ ci-dessus. Au passage, exprimons le développement en fonction de la variable z, et faisons $\_C_0=0$, $\_C_1=1$ pour isoler la solution de l'équation différentielle qui nous intéresse :

```
> sol := eval(sol, {_C[0]=0, _C[1]=1, ln=(x->ln(-x)+I*Pi), w=z-1/8});
```

$$(-.3675525969)\left(\ln\left(z-\frac{1}{8}\right)+2\,\mathrm{i}\,\pi+\left(\frac{-32}{9}-\frac{8}{3}\ln\left(z-\frac{1}{8}\right)-\frac{16}{3}\,\mathrm{i}\,\pi\right)\left(z-\frac{1}{8}\right)\right.$$
$$\left.+\left(\frac{1664}{81}+\frac{320}{27}\ln\left(z-\frac{1}{8}\right)+\frac{640}{27}\,\mathrm{i}\,\pi\right)\left(z-\frac{1}{8}\right)^2\right)$$
$$+(.432914606e-1+1.154700538\,\mathrm{i})\left(\frac{4}{3}-\frac{8}{3}z+\frac{320}{27}\left(z-\frac{1}{8}\right)^2\right)$$

Il s'agit maintenant de déterminer en fonction de n l'expression du coefficient de z^n dans chacun des termes du développement. Tous les *termes* de la dernière ligne sont des polynômes, où ce coefficient est nul pour n assez grand, et donc la contribution de celle ligne est nulle. Les développements de ceux des deux premières lignes sont donnés par la proposition 8.33. Par facilité, au lieu d'expliciter leur calcul, nous faisons appel à la fonction `equivalent` de gdev :

```
> simplify(equivalent(op([1,2], sol), z, n, 3), 'power') assuming n::posint;
```

$$-\frac{8^n}{n}+\frac{1}{3}\frac{8^n}{n^2}-\frac{1}{27}\frac{8^n}{n^3}+O\left(\frac{8^n}{n^4}\right)$$

Transfert. Le premier terme omis dans le développement asymptotique de y au voisinage de $1/8$ est en $w^3\,\ell(w)$. Toujours d'après la proposition 8.33, sa contribution est $O(n^{-4})$. Le théorème 8.31 permet de transférer le résultat à la suite (u_n). On obtient le développement

$$u_n = c\,8^n\left(\frac{1}{n}-\frac{1}{3\,n^2}+\frac{1}{27\,n^3}+O\left(\frac{1}{n^4}\right)\right) \qquad \text{où} \qquad c\approx 0{,}3675525969.$$

Le calcul qui y a mené est complètement rigoureux — à ceci près que, comme déjà indiqué, les bornes pour l'évaluation garantie de c ne sont pas encore implémentées — et entièrement automatisable. On peut bien sûr le pousser à des précisions nettement plus grandes. En quelques secondes, on calcule par exemple que

$$u_n = c\,8^n\left(\frac{1}{n}-\frac{1}{3\,n^2}+\frac{1}{27\,n^3}+\frac{1}{81\,n^4}+\frac{1}{243\,n^5}+\cdots+\frac{3385865210333201}{22876792454961\,n^{20}}+O\left(\frac{1}{n^{21}}\right)\right)$$

$$c\approx 0{,}367552596947861366 34\,(\ldots 160\text{ chiffres}\ldots)38300940123861912929.$$

Identification de la constante. Il se trouve que la constante c que nous avons calculée admet une forme close simple, que la commande `identify` de Maple est capable de « deviner » :

```
> identify(.3675525969);
```

$$\frac{2}{3}\frac{\sqrt{3}}{\pi}$$

Pour reconnaître des valeurs plus compliquées, dix chiffres ne suffisent pas forcément. Plus modestement, notre constante $c = 2/(\pi\sqrt{3})$ peut facilement être confondue avec $e^{-1}\approx 0{,}3679$ — c'est en fait arrivé dans ce cas précis [90, p. 386] ! La précision arbitraire s'avère précieuse tant pour fournir des données plus précises aux algorithmes d'identification de constantes, que pour se convaincre, à défaut d'une preuve, qu'une valeur conjecturée est bien correcte.

Cela s'applique en particulier quand la constante qui apparaît devant le développement de la série génératrice aux singularités dominantes de l'équation différentielle semble nulle. On ne sait pas en général prouver algorithmiquement qu'elle l'est vraiment (voir §3.3.4). Cependant, il reste possible de le vérifier à grande précision, avant d'examiner les contributions des singularités plus éloignées, qui auraient été au-delà de tout ordre sinon.

Chapitre 9

Approximation uniforme sur un segment

« Tu as le regard qui biaise
De tous les doctorants
Et tu deviens obèse
Pris dans ta léthargie

Tu dois finir ta thèse
Tu dois finir ta thèse... »

— Simon Berjeaut, *Le Minotaure* [22]

Ce dernier chapitre représente une version préliminaire d'un travail en commun avec Alexandre Benoit (autre doctorant du projet Algorithms) et Mioara Joldeş (doctorante de l'équipe Arénaire à l'ENS de Lyon). L'essentiel du texte est tiré d'un article commun en préparation [20].

Nous nous intéressons à *l'approximation d'une fonction D-finie par un polynôme* de degré donné, au sens de la norme de la convergence uniforme sur un segment, dans une optique de calcul numérique garanti.

Il est classique que les séries de Tchebycheff tronquées fournissent des approximations polynomiales quasi-optimales. Dans le cas D-fini, les coefficients du développement de Tchebycheff satisfont même une récurrence. Celle-ci ne permet pas de les calculer aussi aisément que les coefficients de Taylor, en raison, entre autres, de la difficulté d'accès aux conditions initiales. Nous montrons comment s'en servir néanmoins pour obtenir des approximations de bonne qualité, accompagnées de *bornes d'erreurs rigoureuses*, le tout en *complexité arithmétique linéaire*. Notre approche se fonde sur une méthode numérique classique due à Clenshaw, combinée à une technique d'arithmétique d'intervalles pour la solution certifiée d'équations différentielles. Quelques résultats intermédiaires sur les développements en série de Tchebycheff de fonctions D-finies sont d'intérêt plus général.

9.1 Introduction

9.1.1 Contexte

Ce chapitre est consacré à l'approximation uniforme garantie des fonctions D-finies sur un segment. Après l'évaluation ponctuelle et à grande précision, les applications envisagées ici concernent plutôt les évaluations répétées à précision modérée. La question qui nous occupe est la suivante.

Question 9.1. Soit y une fonction D-finie, spécifiée comme d'habitude par une équation différentielle linéaire à coefficients polynomiaux accompagnée de conditions initiales convenables. Soit $d \in \mathbb{N}$. Étant donnés y et d, comment calculer les coefficients d'un polynôme

$$p(x) = \sum_{n=0}^{d} c_n T_n(x),$$

écrit dans la base de Tchebycheff (T_n), ainsi qu'une borne R « assez petite » telle que $|y(x) - p(x)| \leqslant R$ pour tout $x \in [-1, 1]$? ♢

Notre première motivation est le problème des évaluations répétées rencontré à une poignée de reprises dans les chapitres précédents. Diverses opérations sur les fonctions mathématiques requièrent de pouvoir évaluer une fonction donnée en de nombreux

points répartis sur un segment, le plus souvent à précision modérée. Outre le tracé de graphes et l'intégration numérique, mentionnons justement calcul de polynômes d'interpolation ou d'approximation d'une fonction donnée.

Une façon standard de répondre à ce besoin consiste à remplacer la fonction par une approximation polynomiale suffisamment précise. Comme par ailleurs l'un des grands axes de ce mémoire est d'étendre le support pour les fonctions D-finies arbitraires dans les systèmes de calcul formel, il est naturel de chercher à calculer des approximations polynomiales de bonne qualité de ces dernières. Des bornes sur les erreurs introduites par l'approximation sont indispensables si l'on veut — toujours dans la lignée des objectifs généraux de cette thèse — utiliser le résultat dans un calcul plus complexe lui-même rigoureux.

Outre une évaluation aisée, les approximations polynomiales offrent une représentation commode des fonctions continues, sur laquelle on peut définir toute une arithmétique, avec addition, multiplication, composition et intégration entre autres opérations. Pour diverses raisons, il est naturel quand on travaille sur un segment d'écrire les polynômes sur la base de Tchebycheff plutôt que sur la base monomiale. En particulier, les troncatures qui interviennent suite aux opérations sur les approximants conservent ainsi de bonnes propriétés d'approximation uniforme. Le logiciel Chebfun de Trefethen *et al.* [231, 78] est un système populaire de calcul numérique fondé sur cette idée. En comparaison de la représentation exacte des fonctions D-finies considérée dans l'essentiel de ce mémoire, ce genre de représentation polynomiale n'est qu'approchée (et parfois moins efficace car beaucoup moins compacte), mais s'applique à une classe de fonctions considérablement plus vaste.

Dans un contexte encore plus général, Epstein, Miranker et Rivlin ont développé un formalisme de calcul sur les fonctions mathématiques[1] appelé « ultra-arithmétique », qui se veut à celles-ci ce que l'arithmétique en virgule flottante est aux nombres réels [85, 86, 139]. Diverses *séries de Fourier généralisées*, en particulier les séries de Tchebycheff, y jouent le rôle des flottants. La définition de l'ultra-arithmétique propose aussi un analogue dans les espaces fonctionnels de l'arithmétique d'intervalles. Les objets de base sont alors des séries tronquées à coefficients intervalles accompagnées de bornes rigoureuses sur les erreurs de troncature. Cette approche de l'arithmétique sur les fonctions a été récemment remise au goût du jour avec l'introduction des « ChebModels » par Brisebarre et Joldeş [41]. Une deuxième motivation derrière la question 9.1 consiste à permettre l'utilisation de fonctions D-finies quelconques aux feuilles des arbres d'expressions que l'on se propose d'évaluer sous forme de ChebModels.

Mais l'attrait principal de l'ultra-arithmétique et des techniques apparentées (voir §6.1.2) est peut-être la possibilité de résoudre rigoureusement des équations fonctionnelles par des méthodes d'inclusion [179, 139, 164, 192]. Les équations différentielles linéaires à coefficients polynomiaux sont parmi les plus simples auxquelles s'applique cette approche. Ce travail vise aussi à contribuer à l'étude de sa *complexité* algorithmique, du point de vue du calcul formel, en prenant comme prototype cette famille de problèmes.

REMARQUE 9.2. Ce chapitre est adapté d'une version *préliminaire* d'un article en préparation [20]. Le caractère inachevé du travail décrit se ressent parfois. On en trouvera d'autres moutures, qui mettent en avant des aspects différents, dans les thèses d'Alexandre Benoit [18] et de Mioara Joldeş [136]. ♢

9.1.2 Données

Dans toute la suite, on fixe une équation différentielle linéaire homogène à coefficients polynomiaux

$$L \cdot y = a_r y^{(r)} + a_{r-1} y^{(r-1)} + \cdots + a_0 y = 0, \qquad a_i \in \mathbb{Q}[x]. \tag{9.1}$$

1. Déjà brièvement mentionné en §6.1.2.

Quitte à effectuer un changement de variable, on recherche une approximation polynomiale d'une solution de (9.1) sur le segment $[-1; 1]$. La norme uniforme sur cet intervalle est notée $\|\cdot\|_\infty$. On suppose aussi que a_r ne s'y annule pas, de sorte que toute solution de (9.1) est analytique sur $[-1; 1]$.

Outre l'opérateur L, on suppose données des conditions initiales à l'origine $y(0), \ldots, y^{(r-1)}(0)$. Une grande partie des résultats du chapitre[2] tient plus généralement en les remplaçant par r conditions au bord

$$\lambda_i(y) = \ell_i, \qquad 1 \leqslant i \leqslant r \tag{9.2}$$

chacune de la forme

$$\lambda_i(y) = \sum_{j=1}^{q} \mu_j y^{(r_j)}(x_j) \tag{9.3}$$

avec $x_j \in [-1; 1]$ et $r_j \leqslant r - 1$. Celles-ci sont alors choisies indépendantes en tant que formes linéaires définies sur $\ker L$, de sorte que la fonction y que l'on étudie soit l'unique solution de (9.1) satisfaisant (9.2). Le cas de conditions initiales données *en-dehors* de l'intervalle de développement peut si nécessaire être ramené à nos hypothèses par l'algorithme de prolongement analytique numérique du chapitre 8.

9.1.3 Modèle de complexité

Sauf mention contraire, nous supposons que les opérations arithmétiques entre nombres réels ou complexes sont effectuées en arithmétique exacte (c'est-à-dire en général rationnelle). La rigueur des calculs n'est pas affectée si l'on remplace l'arithmétique exacte par de l'arithmétique en virgule flottante dans l'algorithme 9.16 et par de l'arithmétique d'intervalles dans l'algorithme 9.32. Cependant, nous n'analysons pas l'effet des erreurs d'arrondi sur la qualité du polynôme p ou de la borne d'erreur B calculés dans ce cas. Dans les cas simples du moins, on peut s'attendre à ce que l'algorithme 9.16 présente une stabilité numérique comparable à celle des méthodes analogues à base de récurrences déroulées « à reculons » (voir à ce sujet Wimp [256]). Nos premières expériences dans ce sens, décrites en §9.6, montrent un comportement numérique satisfaisant. En ce qui concerne l'algorithme 9.28, un passage à l'arithmétique d'intervalles nécessite de petits ajustements.

Pour prendre en compte cette variabilité dans l'arithmétique sur les coefficients, nous évaluons ici la complexité des algorithmes dans le *modèle arithmétique* (cf. §7.2). En d'autres termes, nous attribuons un coût unitaire aux opérations dans $\mathbb{Q}$, et nous négligeons aussi bien les variations du temps de calcul liées à la taille des opérandes que les opérations auxiliaires de contrôle.

Le choix du modèle de complexité arithmétique pour un algorithme où interviennent des calculs numériques multi-précision pourrait surprendre. Cependant, sauf annulations spectaculaires, la hauteur des nombres que nous manipulons est approximativement la même que celle des coefficients de p (à savoir $O(d \log d)$ lorsque ceux-ci sont représentés par des rationnels). Or les opérations arithmétiques usuelles tant sur les rationnels que sur les flottants de hauteur bornée par n sont faisables en complexité binaire $O\big(n\,(\log n)^{O(1)}\big)$ (voir §7.2). On s'attend ainsi à une complexité *binaire* quasi-linéaire en la taille totale du polynôme d'approximation calculé.

9.1.4 Résultats et plan du chapitre

Nous décrivons une méthode de calcul d'approximations polynomiales des fonctions D-finies avec les caractéristiques suivantes. Soit y une fonction D-finie analytique sur le segment $[-1; 1]$, et notons $\|\cdot\|_\infty$ la norme uniforme sur ce segment. Pour un degré d donné, notre méthode renvoie un polynôme p de degré d ainsi qu'une borne rigoureuse et fine R sur la différence $\|y - p\|_\infty$. Le polynôme p est obtenu comme une approximation du développement en série de Tchebycheff de y tronqué à l'ordre d (voir les rappels dans la section suivante).

2. Et vraisemblablement la totalité, moyennant un peu plus de travail.

Sous des hypothèses convenables, la complexité arithmétique du procédé est linéaire en $d + \log(\varepsilon^{-1})$, où ε est lié à la précision de calcul des coefficients de p ainsi de la borne R. Si la fonction y à approcher n'est pas elle-même un polynôme de degré inférieur ou égal à d, on attend « en général », dans un sens volontairement vague[3],

$$\max_x |y(x) - p(x)| \approx \varepsilon \approx 2^{-\Theta(d)}$$

de sorte que les termes d et $\log(\varepsilon^{-1})$ de la borne de complexité sont du même ordre de grandeur.

Mais le développement de Taylor de y à l'origine jusqu'à l'ordre d se calcule aussi en $O(d)$ opérations arithmétiques ! Et les troncatures successives de la série de Taylor constituent, elles aussi, des approximations polynomiales de y d'erreur absolue de l'ordre de 2^{-e} où $e = \Theta(d)$ sur tout compact inclus dans leur disque de convergence. La série de Tchebycheff ne fait pas mieux à cette aune : le temps nécessaire pour obtenir une erreur inférieure à 2^{-e} reste (en général) linéaire en e. En revanche, il ne suffit pas que y soit analytique sur $[-1; 1]$ pour que son développement en série entière à l'origine converge sur ce segment. De façon liée, la constante que cache l'écriture $e = \Omega(d)$ dans le cas des séries de Taylor peut devenir arbitrairement grande quand y varie, même si y *admet* en fait de bien meilleures approximations de degré d. Au contraire, si l'on note $\pi_d(y)$ la série de Tchebycheff de y tronquée à l'ordre d et p^* la *meilleure* approximation de degré d de y, il existe une constante λ, indépendante de y (et dépendant, modérément, de d) telle que $\|y - \pi_d(y)\|_\infty \leqslant \lambda \|y - p_d^*\|_\infty$.

L'objet de notre étude est ainsi d'égaler le coût linéaire (et non seulement *quasi-linéaire*) de l'approximation par séries de Taylor, tout en calculant des approximations dont la qualité est proche de l'optimum, uniformément en y. Dans l'état actuel de ce travail, nous n'énonçons cependant pas de lien précis entre la borne R produite par notre méthode et l'erreur optimale (« minimax ») $\|y - p_d^*\|_\infty$.

Notre méthode procède en deux phases. On commence par calculer un polynôme d'approximation candidat, à partir du développement en série de Tchebycheff de la fonction y. Cette étape fait l'objet de la section 9.3. L'algorithme de calcul du polynôme p rappelle la variante de Fox et Parker [98, Chap. 5] de la méthode de Clenshaw [66]. La clé de la complexité linéaire est une récurrence satisfaite par les coefficients de Tchebycheff, étudiée en §9.2. Sous quelques hypothèses simplificatrices (hypothèses **H1** et **H2** page 188), nous montrons que le polynôme calculé peut être rendu arbitrairement proche du développement de Tchebycheff de y tronqué au degré d. Aucun effort n'est fait pour contrôler rigoureusement l'erreur à ce stade.

Dans un second temps, en section 9.5, on valide le résultat de la première étape par une technique d'inclusion fonctionnelle, de manière à établir une borne sur l'erreur. Cette étape suppose que l'on sait déjà résoudre la question 9.1 appliquée aux *coefficients* de l'équation différentielle qui définit y. On étudie donc auparavant, dans la section 9.4, les développements de Tchebycheff des fractions rationnelles, avec là aussi un algorithme de calcul des coefficients et des bornes sur les restes.

La section 9.6 présente quelques premiers résultats expérimentaux obtenus avec notre implémentation (à l'état de prototype) des algorithmes de ce chapitre. Remarquons aussi que certains résultats intermédiaires sur les développements de Tchebycheff des fractions rationnelles (§9.4) et des fonctions D-finies (§9.2.2-§9.2.4) peuvent être utiles dans un contexte plus large.

9.2 Développement d'une fonction D-finie en série de Tchebycheff

9.2.1 Séries de Tchebycheff

Rappelons[4] que les polynômes de Tchebycheff de première espèce sont les poly-

3. Il est possible de donner des énoncées rigoureux dans ce sens à l'aide de résultats de théorie de l'approximation [47, §4.4], mais nous laissons cela pour une version plus aboutie de ce travail.

nômes $T_n \in \mathbb{Q}[x]$ définis pour tout $n \in \mathbb{Z}$ par la relation

$$T_n(\cos\theta) = \cos(n\,\theta).$$

La famille $(T_n)_{n\in\mathbb{N}}$ forme une suite de polynômes orthogonaux sur le segment $[-1;1]$ relativement au poids

$$w(x) = \frac{1}{\sqrt{1-x^2}}$$

et par conséquent une base hilbertienne de l'espace $L^2(w)$.

Les développements de fonctions $f \in L^2(w)$ sur cette base sont appelées *séries de Tchebycheff*. Au lieu de la notation la plus courante

$$\sum\nolimits'_n u_n T_n = \frac{u_0}{2} T_0 + u_1 T_1 + u_2 T_2 + \cdots$$

nous écrivons les séries de Tchebycheff sous la forme[5]

$$f(x) = \sum_{n=-\infty}^{\infty} c_n T_n(x), \qquad c_{-n} = c_n, \tag{9.4}$$

où les *coefficients de Tchebycheff* $c_n = \frac{1}{2} u_n$ sont donnés par

$$c_n = \frac{1}{\pi} \int_{-1}^{1} \frac{f(x)\,T_n(x)}{\sqrt{1-x^2}}\,\mathrm{d}x, \qquad n \in \mathbb{Z}. \tag{9.5}$$

Ce choix rendra plus transparents le lien entre séries de Tchebycheff et séries de Laurent, ainsi que l'action d'opérateurs de récurrence sur la suite des coefficients c_n — l'un comme l'autre discutés plus bas. Nous notons

$$\pi_d : \quad f \mapsto \sum_{n=-d}^{d} c_n T_n$$

la projection orthogonale sur l'espace des polynômes de degré au plus d.

Supposons maintenant que f est une fonction D-finie, solution de l'équation différentielle

$$L \cdot y = a_r\, y^{(r)} + a_{r-1}\, y^{(r-1)} + \cdots + a_0\, y = 0, \quad a_i \in \mathbb{Q}[x]. \tag{9.6}$$

Comme nous l'avons vu en §3.2.2, la fonction f admet donc un prolongement analytique à tout ouvert simplement connexe $U \subset \mathbb{C}$ qui ne contient pas de point singulier de l'équation. Soit

$$E_\rho = \{x \in \mathbb{C} : |x + \sqrt{x^2-1}| < \rho\} \tag{9.7}$$

le plus grand disque elliptique de foyers en ± 1 ayant cette propriété. Comme les points singuliers de (9.1) sont en nombre fini, on a $1 < \rho \leqslant \infty$. De manière analogue à ceux des séries de Taylor (voir chapitre 3), les coefficients c_n satisfont alors $c_n = O(\alpha^n)$ pour tout réel $\alpha > \rho^{-1}$, et le développement de Tchebycheff (9.4) converge uniformément vers f sur E_r [169, Theorem 5.16].

En posant $x = \cos\theta$ et $z = e^{i\theta}$, on voit facilement que les c_n sont aussi les coefficients du développement en série de Fourier de la fonction $\theta \mapsto f(\cos\theta)$, ou encore de celui en série de Laurent (bi-infinie) de

$$\hat{f} : z \mapsto f\left(\frac{z + z^{-1}}{2}\right)$$

4. Les preuves des résultats résumés ici se trouvent dans les livres de Rivlin [210] ou de Mason et Handscomb [169].

5. Le signe = est un peu abusif : le développement de Tchebycheff ne converge en général qu'au sens L^2. « Nos » fonctions sont analytiques ; la convergence est alors uniforme (et rapide).

au voisinage du cercle unité. La transformation $x = \frac{z+z^{-1}}{2}$ qui à f associe $\hat{f}$ s'appelle transformée de Joukowski inverse. Elle envoie le disque elliptique E_r sur la couronne $A_r = \{z \in \mathbb{C} : \rho^{-1} < |z| < \rho\}$. La formule $T_n(\cos x) = \cos(n\,x)$ se traduit en

$$T_n\left(\frac{z+z^{-1}}{2}\right) = \frac{z^n + z^{-n}}{2}.$$

Soit $\mathcal{C} \subset \mathbb{C}^{\mathbb{Z}}$ l'espace vectoriel des suites bi-infinies $(c_n)_{n\in\mathbb{Z}}$ telles que

$$(\forall n \in \mathbb{N})\,(c_n = c_{-n}) \qquad \text{et} \qquad (\exists \alpha < 1)\,\big(c_n = O(\alpha^n)\big).$$

Ainsi, la suite des coefficients de Tchebycheff d'une fonction f analytique sur un voisinage complexe du segment $[-1\,;1]$ appartient à $\mathcal{C}$. Réciproquement, pour toute suite $c \in \mathcal{C}$, la série de fonctions $\sum_{n=-\infty}^{\infty} c_n\,T_n(x)$ converge uniformément sur (un certain voisinage de) $[-1, 1]$ vers une fonction analytique f.

Les séries de Tchebycheff tronquées sont sur $[-1\,;\,1]$ des approximations *quasi-minimax* de leurs sommes : on a [232, Theorem 16.1]

$$\|f - \pi_d(f)\|_\infty \leqslant \left(\frac{4}{\pi^2}\log(d+1) + 4\right)\|f - p_d^*\|_\infty \tag{9.8}$$

où p_d^* désigne le polynôme de degré au plus d qui minimise $\|f - p\|_\infty$.

Il est vrai que p_d^* lui-même peut en principe être calculé à précision arbitraire par l'algorithme de Remes [47, Chap. 3], mais l'on maîtrise mal la complexité de ce calcul. L'équation (9.8) montre que nous ne perdons pas grand-chose à le remplacer par $\pi_d(f)$. Qui plus est, des approximations plus fines que cette dernière tendent à devenir difficile à valider sans passer par une comparaison avec un polynôme d'approximation intermédiaire de degré plus élevé [51]. Calculer ces approximations intermédiaires fait d'ailleurs partie des besoins qui nous ont conduit à ce travail. Notre choix des séries de Tchebycheff tronquées de préférence à d'autres approximations quasi-minimax jouissant de propriétés analytiques sympathiques, est motivé essentiellement par l'existence d'une relation de récurrence sur les c_n lorsque f est une fonction D-finie.

9.2.2 La récurrence de Tchebycheff

Les polynômes T_n satisfont la récurrence

$$2\,x\,T_n(x) = T_{n-1}(x) + T_{n+1}(x), \tag{9.9}$$

ainsi que la relation mixte différentielle et aux différences

$$2\,(1-x^2)\,T_n'(x) = n\,(T_{n+1}(x) - T_{n-1}(x)) \tag{9.10}$$

qui se traduit par la formule d'intégration

$$2\,n\,c_n' = c_{n-1} - c_{n+1}, \qquad \sum c_n' T_n = \big(\sum c_n T_n\big)'.$$

C'est de ces égalités que découle l'ingrédient-clé des travaux développés ici, à savoir que les coefficients du développement de Tchebycheff d'une fonction D-finie vérifient une récurrence linéaire à coefficients polynomiaux.

Cela fut observé par Fox et Parker [97, 98] dans des cas particuliers, puis démontré en général par Paszkowski [198]. Lewanowicz a exploré les propriétés de cette récurrence et développé des généralisations à d'autres bases de polynômes orthogonaux dans une série d'articles débutant en 1976 (voir en particulier [157, 158]). Geddes [106] a été le premier à étudier sa détermination automatique dans un logiciel de calcul formel. Comme indiqué en introduction de ce mémoire, les séries de Tchebycheff D-finies intéressaient déjà les mathématiciens auparavant, en lien notamment avec des questions de calcul numérique [153, 154, 66]. Une fois isolée la notion de D-finitude, ces

séries sont étudiées du point de vue du calcul formel dans les thèses de Rebillard [207] puis de Benoit [18].

Le théorème suivant récapitule quelques résultats relatifs à cette récurrence, tirés de travaux existants [198, 157, 158, 207, 21] et étendus pour nos besoins. Rappelons que $\mathbb{Q}(n)\langle S, S^{-1}\rangle$ est l'anneau de polynômes de Laurent tordus sur $\mathbb{Q}(n)$ en l'indéterminée S, soumis aux règles de commutation

$$S\,\lambda = \lambda\,S \quad (\lambda \in \mathbb{Q}), \qquad S\,n = (n+1)\,S, \tag{9.11}$$

et $\mathbb{Q}[n]\langle S, S^{-1}\rangle$ son sous-anneau formé des éléments polynomiaux en n. Les éléments de ce dernier s'identifient à des opérateurs de récurrence sur $\mathbb{C}^{\mathbb{Z}}$ à travers leur action à gauche évidente. Rappelons aussi que L est l'opérateur différentiel qui apparaît dans l'équation (9.1).

Théorème 9.3. [198, 157, 158, 207, 21] Soient u, v des fonctions analytiques sur un voisinage complexe du segment $[-1;1]$, admettant les développements de Tchebycheff

$$u(x) = \sum_{n=-\infty}^{\infty} u_n\,T_n(x), \quad v(x) = \sum_{n=-\infty}^{\infty} v_n\,T_n(x).$$

Il existe alors deux opérateurs aux différences $P, Q \in \mathbb{Q}[n]\langle S, S^{-1}\rangle$ avec les propriétés suivantes.

(a) L'équation différentielle $L \cdot u(x) = v(x)$ est satisfaite si et seulement si l'on a

$$P \cdot (u_n) = Q \cdot (v_n). \tag{9.12}$$

(b) L'opérateur au membre gauche, P, est de la forme

$$P = \sum_{k=-s}^{s} b_k(n)\,S^k,$$

où $s = r + \max_i(\deg a_i)$ et $b_{-k}(-n) = -b_k(n)$ pour tout k.

(c) En posant

$$\delta_r(n) = 2^r \prod_{i=-r+1}^{r-1} (n-i), \qquad I = \frac{1}{2\,n}\,(S^{-1} - S), \tag{9.13}$$

on a $Q = Q_r = \delta_r(n) I^r$ (formellement, dans $\mathbb{Q}(n)\langle S, S^{-1}\rangle$). En particulier, Q ne dépend que de r et vérifie la même propriété de symétrie que P. ♢

Observons en passant que I, tel qu'il est défini dans l'équation (9.13), peut s'interpréter comme un opérateur qui envoie les suites symétriques $(u_{|n|})_{n\in\mathbb{Z}}$ vers les suites indéfinies en zéro $(u_n)_{n\in\mathbb{Z}\setminus\{0\}}$. Ainsi, une manière de reformuler le point central du théorème 9.3, un peu vague mais qui peut peut-être aider l'intuition, serait quelque chose comme : « on a $(\int)^r L \cdot u = w$ si et seulement si $\delta_r(n)\,P \cdot u = w$, aux constantes d'intégration près ».

Démonstration. Supposons $L \cdot u = 0$. Benoit et Salvy [21, Theorem 1] donnent une preuve simple de l'existence d'opérateurs $P, Q \in \mathbb{Q}(n)\langle S, S^{-1}\rangle$ satisfaisant (9.12). La forme de P et Q résulte de la construction explicite qui en est donnée dans ce même article [21, §4.1], sur la base de l'algorithme de Paszkowski. Plus précisément, on obtient une récurrence de la forme prescrite en multipliant les deux membres de [21, Eq. (17)] par $\delta_r(n)$. Elle est à coefficients polynomiaux car $\delta_r(n)\,I^r \in \mathbb{Q}(n)\langle S, S^{-1}\rangle$ (et les q_i qui apparaissent dans la formule [21, Eq. (17)] sont des polynômes), observation démontrée en détail dans la thèse de Rebillard [207, §4.1], de même que toutes les assertions du point (b). Notons que si les références citées sont sans doute les plus accessibles avec le contexte de cette thèse, plusieurs de ces résultats remontent dans un langage un peu différent à Paszkowski [198] et Lewanowicz [157, 158].

Il reste à prouver le sens « si » de l'équivalence. Considérons deux suites $u, v \in C$ telles que $P \cdot u = Q \cdot v$, et soit $y \in C$ la suite définie par

$$L \cdot u(x) = \sum_{n=-\infty}^{\infty} y_n T_n(x).$$

D'après ce qui précède, on a donc $P \cdot u = Q \cdot y$, mais alors $Q \cdot v = Q \cdot y$, et l'on conclut que $u = y$ par le lemme suivant. □

LEMME 9.4. Restreint à l'espace $\mathcal{C}$ des suites bi-infinies symétriques à convergence exponentielle, l'opérateur Q défini au théorème 9.3 est injectif. ♢

DÉMONSTRATION. Avec les notations du théorème 9.3, montrons par récurrence sur $r \geqslant 1$ que

$$(v \in \mathcal{C}) \wedge \big(|n| \geqslant r \Rightarrow (Q_r \cdot v)_n = 0\big) \Rightarrow v = 0. \tag{9.14}$$

On a $(\ker Q_1) \cap \mathcal{C} = \{0\}$ puisque toute suite de $\mathcal{C}$ converge vers zéro quand $n \to \pm\infty$. Supposons (9.14) vérifiée, et soit $v \in \mathcal{C}$ une suite telle que

$$(Q_{r+1} \cdot v)_n = 0 \qquad \text{pour} \qquad |n| \geqslant r+1.$$

Soit $w = Q_r \cdot v$. On observe qu'alors $w \in \mathcal{C}$. Comme $r \geqslant 1$, on a

$$\begin{aligned} n\, Q_{r+1} &= \delta_{r+1}(n)\,(S^{-1} - S)\, I^r \\ &= \Big((n+r)\,(n+r-1)\, S^{-1}\, \delta_r(n) - (n-r)\,(n-r+1)\, S\, \delta_r(n) \Big)\, I^r \\ &= \Big((n+r)\,(n+r-1)\, S^{-1} - (n-r)\,(n-r+1)\, S \Big)\, Q_r \end{aligned}$$

d'où, pour $|n| \geqslant r+1$,

$$(n+r)\,(n+r-1)\, w_{n-1} = (n-r)\,(n-r+1)\, w_{n+1}. \tag{9.15}$$

À moins que (w_n) ne soit ultimement nulle, il s'ensuit que $w_{n+1}/w_{n-1} \to 1$ lorsque $n \to \infty$, or cela est incompatible avec l'observation que $w \in \mathcal{C}$. Par conséquent, on a $w_n = 0$ pour $|n|$ grand, et donc, en utilisant à nouveau (9.15), $w_n = 0$ dès que $|n| \geqslant r$. On conclut en appliquant l'hypothèse (9.14). □

Une manière facile à présenter de calculer une récurrence de la forme (9.12) consiste à effectuer le changement de variable $x = \frac{1}{2}(z + z^{-1})$ dans l'équation différentielle, puis à calculer par l'algorithme classique (proposition 3.7) une récurrence sur les coefficients de Laurent d'une solution de l'équation obtenue. Curieusement, cette méthode, sans nul doute connue des spécialistes, ne semble pas apparaître explicitement dans la littérature.

Benoit et Salvy [21] donnent une présentation unifiée de divers autres algorithmes de calcul de cette récurrence — dont l'algorithme de Paszkowski mentionné dans la preuve — en les interprétant comme différentes manières de réaliser la substitution

$$x \mapsto \frac{1}{2}(S + S^{-1}), \qquad \partial \mapsto (S - S^{-1})^{-1}\,(2\,n)$$

dans une algèbre à division non commutative convenable. Ils montrent que tous ces algorithmes calculent le même opérateur P (en l'absence de singularités de l'équation différentielle sur $[-1; 1]$, ce qui correspond au cas qui nous intéresse).

DÉFINITION 9.5. À la suite de Rebillard, nous appellerons la relation (9.12) produite par l'algorithme de Paszkowski ou les algorithmes équivalents la *récurrence de Tchebycheff* associée à l'équation différentielle (9.1). ♢

Dans la suite, nous ne considèrerons en général que des récurrences de Tchebycheff associées à des équations homogènes, qui sont elles-mêmes homogènes.

Remarque 9.6. D'après le théorème 9.3(b) et avec ses notations, on a pour toute suite (u_n) :

$$\forall n, \quad \sum_k b_k(n)\, u_{n+k} = -\sum_k b_{-k}(-n)\, u_{n+k} = -\sum_k b_k(-n)\, u_{-n-k}$$

c'est-à-dire $P \cdot (u_n)_{n\in\mathbb{Z}} = -P \cdot (u_{-n})_{n\in\mathbb{Z}}$. En particulier, si (u_n) est solution de $P\cdot(u_n) = 0$, alors (u_{-n}) l'est aussi. Toute solution (u_n) donne donc naissance à une solution symétrique $(u_n + u_{-n})$. N'importe quelle solution n'est pas pour autant symétrique. ♢

9.2.3 Solutions de la récurrence de Tchebycheff

Plusieurs difficultés se présentent lorsque l'on cherche à recourir à la récurrence de Tchebycheff pour calculer les coefficients de Tchebycheff d'une fonction.

La première est liée aux conditions initiales. Peut-être est-il utile ici de comparer la situation avec celle des séries entières. Dans les chapitres précédents, nous n'avons eu aucune peine à exprimer les premiers coefficients *de Taylor* de y pour initialiser la récurrence correspondante. En revanche, les coefficients de Tchebycheff $c_0, c_1, \ldots$ que l'on serait tenté d'adopter comme conditions initiales à la récurrence de Tchebycheff ne sont pas reliés de façon simple à des conditions initiales ou des conditions au bord naturelles de l'équation différentielle. En particulier, comme le montre le théorème 9.3 ci-dessus, l'ordre $2\,s$ de la récurrence de Tchebycheff est strictement plus élevé (sauf cas dégénérés) que celui de l'équation différentielle. Il nous faut dont obtenir d'une manière ou d'une autre « plus de conditions intiales que nous n'en avons naturellement sous la main »[6].

En second lieu, et toujours à la différence de ce qu'il se passe avec les séries entières, les coefficients de tête et de queue de la récurrence (9.12) peuvent s'annuler pour des valeurs arbitrairement grandes de n, bien que l'équation différentielle (9.1) soit non singulière sur $[-1;1]$. Rappelons que les zéros de $b_s(n-s)$ s'appellent *singularités de tête* de (9.12), et ceux de $b_{-s}(n+s)$ *singularités de queue*. Dans le cas d'une récurrence de Tchebycheff, l'ensemble des singularités de tête et celui des singularités de queue sont opposés.

Exemple 9.7. Pour tout $k \in \mathbb{Z}$, la récurrence de Tchebycheff associée à l'équation différentielle $y''(x) + x\,y'(x) + k\,y(x) = 0$, à savoir

$$(n+1)\,(n+k-1)\,u_{n-2} + 2\,n\,(2\,n^2+k-1)\,u_n - (n-1)\,(n-k+2)\,u_{n+2} = 0$$

présente une singularité de tête en $n = k-2$. ♢

On dispose néanmoins d'un contrôle partiel sur les singularités.

Proposition 9.8. Avec les notations du théorème 9.3, les coefficients de la récurrence de Tchebycheff donnée par l'algorithme de Paszkowski (ou tout autre algorithme équivalent) satisfont les relations

$$b_{j-i}(-j) = -b_{j+i}(-j), \qquad |j| \leqslant r-1, \quad i \in \mathbb{N}, \tag{9.16}$$

où l'on convient que $b_k = 0$ lorsque $|k| > s$. ♢

6. Il reste tout de même que les coefficients de Tchebycheff sont tous combinaisons linéaires à coefficients rationnels, donnés par la récurrence, d'un nombre fini d'intégrales qu'il est envisageable par exemple de calculer numériquement. La question de la complexité à laquelle on peut aboutir par cette voie pour calculer à grande précision les séries de Tchebycheff est intéressante, et nous espérons y revenir dans un prochain travail.

DÉMONSTRATION. On raisonne par récurrence sur r. Lorsque $j=0$, l'assertion (9.16) se réduit à $b_{-i}(0)=-b_i(0)$, ce qui découle du point (b) du théorème 9.3. Cela démontre en particulier la proposition pour $r=1$.

Soit maintenant $r\geqslant 2$, et supposons les égalités (9.16) établies pour L d'ordre $r-1$. Posons $L=\hat{L}+\partial^r p_r(x)$ où $p_r\in\mathbb{Q}[x]$ et $\hat{L}$ est un opérateur différentiel d'ordre $r-1$. En écrivant $\hat{P}=\sum_{k\in\mathbb{Z}}\hat{b_k}(n)\,S^k$ l'opérateur de récurrence de Tchebycheff associé à $\hat{L}$, on a [21]

$$\delta_r(n)^{-1}\,P=I\,\delta_{r-1}(n)^{-1}\,\hat{P}+p_r\Big(\tfrac{1}{2}\,(S+S^{-1})\Big) \tag{9.17}$$

où le dernier terme se comprend comme l'évaluation en $x=\frac{1}{2}\,(S+S^{-1})$ du polynôme p_r. D'après les règles de commutation (9.11), on a

$$I\,\delta_{r-1}(n)^{-1}=\big(n\,\delta_r(n)\big)^{-1}\Big((n-r+2)\,(n-r+1)\,S^{-1}-(n+r-2)\,(n+r-1)\,S\Big)$$

donc la relation (9.17) se réécrit

$$\begin{aligned}P=\frac{1}{n}\sum_k\ &\big((n-r+2)\,(n-r+1)\,\hat{b_{k+1}}(n-1)-(n+r-2)\,(n+r-1)\,\hat{b_{k-1}}(n+1)\big)\,S^k\\&+\delta_r(n)\,p_r(S+S^{-1}).\end{aligned}$$

Le cas $j=0$ étant déjà réglé, on peut supposer $0<|j|<r$. Comme $\delta_r(-j)=0$ et comme p_r est un polynôme, il vient en extrayant le coefficient de S_k dans l'égalité précédente puis en évaluant en $n=-j$:

$$-j\,b_k(-j)=(j+r-2)\,(j+r-1)\,\hat{b_{k+1}}(-j-1)-(j-r+2)\,(j-r+1)\,\hat{b_{k-1}}(-j+1).$$

Or on a $b_{j+i}(-j)=-b_{j+i}(-j)$ pour $|j|<r-1$ par hypothèse de récurrence, tandis que les termes en $\hat{b_{k\pm1}}$ s'annulent respectivement pour $j=\mp(r-1)$ et $j=\mp(r-2)$. On obtient donc dans tous les cas $b_{j-i}(-j)=-b_{j+i}(-j)$. □

NOTATION 9.9. Dans la suite, $P\cdot y=0$ est la récurrence de Tchebycheff associée à l'équation différentielle $L\cdot y=0$, et son demi-ordre est noté s. ♢

Cette observation aura de lourdes conséquences sur la structure des solutions d'une récurrence de Tchebycheff, *via* le corollaire suivant.

COROLLAIRE 9.10. Si $(u_{|n|})_{n\in\mathbb{Z}}$ est une suite symétrique, on a $(P\cdot u)_n=0$ pour tout $|n|<r$. ♢

DÉMONSTRATION. On a

$$(P\cdot u)_n=\sum_{k\in\mathbb{Z}}b_k(n)\,u_{n+k}=\sum_{i\in\mathbb{Z}}b_{i-n}(n)\,u_i$$

or la proposition 9.8 avec $j=-n$ et $|n|<r$ entraîne

$$\sum_{i\in\mathbb{Z}}b_{i-n}(n)\,u_i=-\sum_{i\in\mathbb{Z}}b_{-i-n}(n)\,u_i=-\sum_{i\in\mathbb{Z}}b_{i-n}(n)\,u_i$$

autrement dit $(P\cdot u)_n=-(P\cdot u)_n$. □

Troisième difficulté, et non la moindre, les récurrences de Tchebycheff comportent des solutions divergentes, qui ne correspondent pas à des développements de solutions de l'équation différentielle dont elles sont issues (cf. Rebillard [207, chap. 5]).

EXEMPLE 9.11. La récurrence de Tchebycheff associée à l'équation $y'=y$ s'écrit

$$(P\cdot u)_n=u\,(n+1)+2\,n\,u\,(n)-u\,(n-1)=0.$$

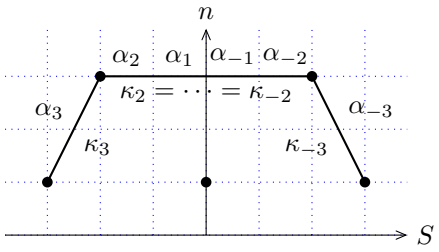


Figure 9.1. Forme du polygone de Newton d'une récurrence de Tchebycheff.

Les suites $(I_\nu(1))_{\nu\in\mathbb{Z}}$ et $(K_\nu(1))_{\nu\in\mathbb{Z}}$, où I et K sont les fonctions de Bessel modifiées, forment une base de solutions de la récurrence. La première est la suite des coefficients de Tchebycheff de la fonction exponentielle ; elle décroît en $\Theta(2^{-n}\,n!^{-1})$. La seconde diverge comme $\Theta(2^{n-1}\,(n-1)!)$. ♢

9.2.4 Solutions convergentes et solutions divergentes

Considérons le polygone de Newton de l'opérateur de récurrence P, tel que défini en section 3.4.2 (voir figure 9.1). Soient

$$\alpha_s, \alpha_{s-1}, ..., \alpha_1, \alpha_{-1}, ..., \alpha_{-s+1}, \alpha_s$$

toutes les racines des équations caractéristiques correspondantes, comptées au besoin avec multiplicités, les arêtes étant parcourues de gauche à droite et les racines des équations caractéristiques d'une même arête par module croissant. Soient

$$-\kappa_s, -\kappa_{s-1}, ..., -\kappa_1, \kappa_1, ..., \kappa_{-s+1}, \kappa_{-s}$$

les pentes des arêtes correspondantes. (Chaque pente est répétée dans la liste un nombre de fois égal à la longueur de la projection sur l'axe des abscisses de l'arête correspondante, et l'on a $\kappa_s \leqslant \kappa_{s-1} \leqslant \cdots \leqslant \kappa_{-s}$.)

D'après le théorème de Perron-Kreuser (théorème 3.25), l'opérateur P annule $2\,s$ suites définies à partir d'un certain rang, linéairement indépendantes, de comportement asymptotique à l'infini décrit par les couples (κ_i, α_i). Il est plus commode ici de donner un énoncé précis en termes de germes de solutions. Rappelons la définition.

Définition 9.12. On appelle *germe de suite* au voisinage de $+\infty$ une classe d'équivalence de suites $(u_n)_{n\geqslant N}$ définies à partir d'un certain rang modulo identification des suites qui coïncident sur l'intersection de leurs domaines de définition. On appelle *germe de solution* d'une récurrence linéaire un germe de suite dont les représentants satisfont la récurrence sur leur domaine de définition. ♢

Dans le cas d'une récurrence à coefficients polynomiaux, les singularités étant en nombre fini, les germes de solution en $+\infty$ s'identifient aux solutions définies à partir de n'importe quel rang N dépassant la plus grande singularité de tête de la récurrence. L'espace des germes de solutions en $+\infty$ est donc de dimension égale à l'ordre de la récurrence. Le comportement asymptotique d'un germe de solution en $+\infty$ est défini sans ambiguïté.

Le résultat du théorème de Perron-Kreuser est que la récurrence $P\cdot u = 0$ admet une base $u_s, ..., u_1, u_{-1}, ..., u_{-s}$ de germes de solutions au voisinage de $+\infty$ tels que

$$\limsup_{n\to\infty} \left|\frac{u_{i,n}}{n!^{\kappa_i}}\right|^{1/n} = |\alpha_i|.$$

Les propriétés de symétrie des coefficients de P se traduisent sur le polygone de Newton.

Proposition 9.13. Les pentes et racines des équations caractéristiques du polygone de Newton de P vérifient $\kappa_{-i} = -\kappa_i$ et $|\alpha_{-i}| = |\alpha_i|^{-1}$ pour tout i. En outre, l'équation

caractéristique associée à l'arête horizontale (si celle-ci existe) n'a pas de racine de module 1. ♢

DÉMONSTRATION. D'après le théorème 9.3(b), les coefficients b_k de P satisfont $b_{-k}(n) = -b_k(-n)$. Le polygone de Newton est donc symétrique par rapport à son axe vertical, et $\kappa_{-i} = -\kappa_i$ pour tout i. Notons $A_k = (k, \deg b_k)$ pour tout k, et soit $E_i = [A_{g(i)}, A_{d(i)}]$ l'arête de pente κ_i du polygone de Newton. L'équation caractéristique de E_i s'écrit

$$\begin{aligned}\chi_i(\alpha) &= \sum_{k:\,A_k\in E_i} \mathrm{lc}(b_k)\,\alpha^{k-g(i)} = \sum_{k:\,A_k\in E_i} (-1)^{1+\deg b_k}\,\mathrm{lc}(b_{-k})\,\alpha^{k-g(i)} \\ &= \pm \sum_{k:\,A_k\in E_{-i}} \mathrm{lc}(b_k)\,\Big((-1)^{\kappa_i}\,\alpha\Big)^{-k-g(i)} = \pm\alpha^{g(i)-g(-i)}\,\chi_{-i}\Big((-1)^{-\kappa_i}\,\alpha^{-1}\Big)\end{aligned}$$

puisque $\deg b_k - \deg b_{g(i)} = \kappa_i\big(k - g(i)\big)$ lorsque $A_k \in E_i$. On en déduit les relations $|\alpha_{-i}| = |\alpha_i|^{-1}$.

La preuve que $\kappa_i = 0 \Rightarrow |\alpha_i| \neq 1$ est similaire à celle du lemme 5.12 page 85. En bref, on vérifie que sous le changement de variable $x = \frac{1}{2}(z + z^{-1})$,

$$\left(\frac{\mathrm{d}}{\mathrm{d}x}\right)^k = \left(\frac{2}{z - z^{-1}}\right)^k \theta^k + (\text{termes de degré en}\,\theta\,\text{strictement plus petit})$$

où $\theta = z\,(\mathrm{d}/\mathrm{d}z)$. Par conséquent, le coefficient de degré maximal en n dans l'opérateur de récurrence de Tchebycheff associée à l'équation (9.1) (qui est bien défini puisque la commutation entre z et θ, ou S et n, préserve les degrés) est $a_r\big(\frac{1}{2}(S + S^{-1})\big)$. Ce polynôme en S, s'il n'est pas constant, n'est autre que l'équation caractéristique de l'arête horizontale du polygone de Newton. Or a_r ne s'annule pas sur $[-1; 1]$, donc $a_r\big(\frac{1}{2}(\alpha + \alpha^{-1})\big)$ ne s'annule pas sur le cercle unité. □

Ainsi, la base de solutions au voisinage de $+\infty$ donnée par le théorème de Perron-Kreuser est constituée de s solutions *convergentes* u_1, ..., u_s et de s solutions *divergentes* $u_{-1}, ..., u_{-s}$. En particulier, l'espace des germes à l'infini de solutions convergentes de $P \cdot u = 0$ est de dimension s.

9.3 Calcul des coefficients d'un polynôme d'approximation

9.3.1 L'algorithme de Clenshaw revisité

Comment calculer par récurrence, au moins approximativement, les coefficients du développement de Tchebycheff d'une fonction D-finie malgré les difficultés mentionnées plus haut ? L'algorithme que nous proposons est une version plus systématique d'une méthode due à Clenshaw[7] en 1957 [66], reformulée pour mettre en évidence le rôle (déjà observé par Fox et Parker [97, 98]) qu'y joue la récurrence de Tchebycheff. Tant la méthode originale de Clenshaw que notre algorithme sont à rapprocher de la méthode de Miller pour le calcul d'une solution minimale d'une récurrence et de ses généralisations [26, 256].

L'idée de Miller est de calculer les coefficients $u_N, u_{N-1}, ..., u_0$ d'une suite récurrente linéaire « à reculons », à partir de conditions initiales prises au hasard au voisinage de $N \gg 0$. Intuitivement, dans les coefficients u_0, u_1, ..., u_n ($n < N$) ainsi définis, c'est alors le plus rapidement *décroissant* des comportements possibles pour une solution de la récurrence qui domine. Les termes calculés sont donc proches de ceux d'une solution minimale de la récurrence — alors même que le comportement asymptotique de la suite (u_n) lorsque $n \to \infty$ correspond quant à lui à la croissance la

7. À ne pas confondre avec ce que l'on appelle habituellement *l'algorithme de Clenshaw*, à savoir une méthode d'évaluation des polynômes écrits sur la base de Tchebycheff analogue à celle de Horner pour la base monomiale [65].

plus importante (voir figure 9.2). L'intérêt est double : premièrement, cette méthode est bien plus stable numériquement que le calcul de $u_0, u_1, ...$ pour n croissant — qui serait au contraire parasité par une contribution issue de la solution dominante après la moindre erreur d'arrondi. Deuxièmement, on parvient ainsi à caractériser et approcher la solution minimale *via* son comportement asymptotique, sans avoir besoin de connaître au départ les conditions initiales correspondantes.

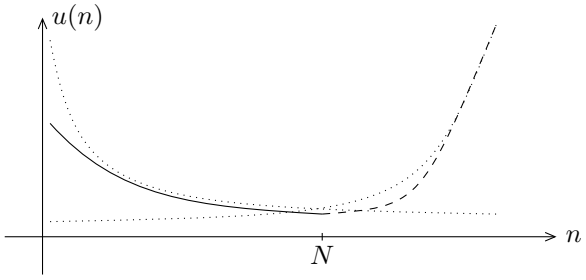


Figure 9.2. Allure stylisée d'une solution test dans un algorithme « à la Miller », comparée à celles d'une solutions « convergente » et d'une solution « divergente ».

Plus généralement, si l'on calcule ainsi s solutions test linéairement indépendantes, on s'attend à ce que leurs restrictions à $[\![0,\, n]\!]$ pour $n < N$ engendrent un espace vectoriel « proche » de celui qu'on obtiendrait avec « les s plus convergentes » des solutions de base données par le théorème de Perron-Kreuser.

Afin de préciser cela dans le cas de la récurrence de Tchebycheff, poursuivons son étude. Soient

$$\boldsymbol{S} = \{n \geqslant s : b_{-s}(n) = 0\},$$

(ce sont les singularités de queue de P à un décalage près) et $k = |\boldsymbol{S}|$. Soit

$$E = \{(u_{|n|})_{n\in\mathbb{Z}} : (n \in \mathbb{N} \setminus \boldsymbol{S}) \Rightarrow (P \cdot u)_n = 0\}$$

l'ensemble des suites *symétriques* dont la *restriction aux indices positifs* est annulée par P, *sauf* peut-être en les indices correspondant à une singularité de queue. Comme, d'après la remarque 9.6,

$$P \cdot (u_n)_{n\in\mathbb{Z}} = -(P \cdot (u_{-n})_{n\in\mathbb{Z}}),$$

tout $u \in E$ satisfait en fait aussi $(P \cdot u)_n = 0$ pour $-n \in \mathbb{N} \setminus \boldsymbol{S}$.

Un élément u de E est caractérisé par la donnée des u_n pour $n + s \in \boldsymbol{S}$ accompagnés d'un germe à l'infini de solution. De façon équivalente, il est caractérisé par $(u_n)_{n\in\boldsymbol{I}}$ où

$$\boldsymbol{I} = (\boldsymbol{S} - s) \cup [\![N, N + 2\,s - 1]\!]$$

pour N suffisamment grand. (En effet, les coefficients u_n pour $n \in \mathbb{N} \setminus (\boldsymbol{S} - s)$ sont alors déterminés par la récurrence, et ceux pour $n < 0$ par symétrie.)

Fait 9.14. La dimension de E est donc exactement $2\,s + k$. ♢

Une suite $u \in E$ est solution (symétrique) de $P \cdot u = 0$ si et seulement si $(P \cdot u)_n = 0$ pour $n \in \boldsymbol{S}$ ainsi que pour $|n| < s$. Mais les conditions $(P \cdot u)_n = 0$ pour $|n| < r$ sont triviales en vertu du corollaire 9.10 : ainsi, $u \in E$ est solution si et seulement si

$$(P \cdot u)_n = 0, \qquad n \in [\![r, s - 1]\!] \cup \boldsymbol{S}. \tag{9.18}$$

Soit $d \leqslant s - r + k$ le rang du système des formes linéaires (9.18) sur l'espace E. La dimension de l'espace des solutions symétriques de la récurrence $P \cdot u = 0$ est

$$\dim E - d = 2\,s + k - d \geqslant s + r.$$

De même, une suite $u \in E$ convergente en $\pm\infty$ est caractérisée par les u_n, $n+s \in \boldsymbol{S}$ ainsi qu'un germe convergent de solution à l'infini, et solution si elle satisfait en outre les contraintes (9.18). Soit $d_{\text{cvg}} \leqslant d$ le rang du système (9.18) restreint à $E \cap \mathcal{C}$. La dimension des solutions convergentes symétriques de $P \cdot u = 0$ s'écrit alors

$$\dim (E \cap \mathcal{C}) - d_{\text{cvg}} = s + k - d_{\text{cvg}}.$$

Or on sait par ailleurs, d'après le théorème 9.3, que cette dimension est égale à r. Ainsi

$$r = s + k - d_{\text{cvg}} \geqslant s + k - d \geqslant r$$

et donc $d = d_{\text{cvg}} = s + k - r$. Récapitulons.

PROPOSITION 9.15. Soient une équation différentielle homogène d'ordre r sans singularité sur le segment $[-1;1]$, et $P \cdot u = 0$ la récurrence de Tchebycheff associée, d'ordre noté $2\,s$.

(a) L'espace des germes à l'infini de solutions de $P \cdot u = 0$ est de dimension $2\,s$.
(b) L'espace des germes à l'infini *convergents* de solutions de $P \cdot u = 0$ est de dimension s.
(c) L'espace des suites $u = (u_n)_{n \in \mathbb{Z}}$ *symétriques* ($u_{-n} = u_n$) solutions de $P \cdot u = 0$ est de dimension $s + r$, et caractérisé parmi les éléments de E par le système (9.18).
(d) L'espace des suites *symétriques convergentes* solutions de $P \cdot u = 0$ est de dimension r, et caractérisé parmi les éléments de $E \cap \mathcal{C}$ par le système (9.18). ♢

Ainsi, l'espace E se décompose en la somme directe de

(i) l'espace des suites de coefficients de Tchebycheff des solutions de l'équation différentielle, de dimension r ;
(ii) un supplémentaire de ces dernières dans les solutions symétriques de la récurrence de Tchebycheff, de dimension s ;
(iii) un supplémentaire des mêmes dans l'espace $\mathcal{C}$ des suites symétriques à convergence exponentielle, de dimension $s - r + k$.

La quatrième composante que l'on pourrait attendre, à savoir un supplémentaire dans E de la somme des solutions et des suites convergentes, est nulle. Toute suite $u \in E$ s'écrit comme somme d'une solution symétrique de $P \cdot v = 0$ et d'une suite symétrique convergente. On pourrait raffiner en distinguant les suites nulles à partir d'un certain rang des autres suites convergentes.

DÉMONSTRATION. Le point (a) est évident. Le point (b) provient de l'examen du polygone de Newton de l'opérateur de récurrence (§9.2.4). La dimension de l'espace des solutions symétriques convergentes de $P \cdot u = 0$ est donnée par le théorème 9.3 ; le point (d) s'en déduit d'après la discussion précédente, et le point (c) résulte de ce dernier et du corollaire 9.10. □

À la lumière de la description précédente des solutions, l'algorithme 9.16 déroule « à reculons » $s + k$ solutions test linéairement indépendantes de la récurrence de Tchebycheff associée à l'équation (9.1). Il en cherche ensuite une combinaison linéaire qui satisfait les conditions (9.18) caractéristiques des solutions symétriques, ainsi que les conditions au bord (9.2). Il prend en entrée à la fois le degré d du polynôme recherché, et un paramètre N indiquant à partir de quel rang calculer les solutions test. Nous étudierons plus loin la façon dont la qualité de l'approximation qu'il renvoie évolue avec N. En pratique, adopter simplement $N = d + s$ donne des résultats satisfaisants.

PROPOSITION 9.17. L'algorithme 9.16 s'exécute en $O(N)$ opérations. □

La preuve de convergence en section suivante (proposition 9.19) relie, dans les cas simples, le choix de N à la qualité de l'approximation de $\pi_d(y)$ par p.

Algorithme 9.16. Clenshaw($L, \{\lambda_i(y)=\ell_i\}_{i=1}^r, d, N$)

Entrée. Un opérateur $L\in\mathbb{Q}[x]\langle\partial\rangle$ d'ordre r et des conditions au bord $\lambda_i(y)=\ell_i$ satisfaisant les hypothèses énoncées en §9.1.2. Deux entiers d, N avec $N\geqslant d>s$, où s est le demi-ordre de la récurrence de Tchebycheff associée à L.

Sortie. Les coefficients $\tilde{y}_n$ d'un polynôme $\tilde{y}(x)=\sum_{n=-d}^{d}\tilde{y}_n T_n(x)$, ou **échec**.

1 calculer l'opérateur de récurrence $P=\sum_{k=-s}^{s} b_k(n)\,S^k$ associé à L

2 $\boldsymbol{S}:=\{n\in\mathbb{N}:(s\leqslant n\leqslant N)\wedge(b_{-s}(n)=0)\}$

3 $\boldsymbol{I}:=\boldsymbol{S}\cup[\![N, N+s-1]\!]$

4 pour $i\in\boldsymbol{I}$

5 pour n décroissant de $N+s-1$ à s

6 calculer $t_{i,n-s}$ par la relation de récurrence $(P\cdot t)_n=0$ (si $n\notin\boldsymbol{I}$) ou les conditions initiales

$$\begin{cases} t_{i,i-s}=1 \\ t_{i,n-s}=0, \quad n\in\boldsymbol{I}\setminus\{i\} \end{cases}$$

7 poser $\tilde{y}_n=$ pour $|n|<N$ et $\tilde{y}_n=0$ sinon

8 $\tilde{y}(x):=\displaystyle\sum_{n=-N}^{N}\tilde{y}_n T_n(x)$, où $\tilde{y}_n:=\sum_{i\in\boldsymbol{I}}\eta_i\,t_{i,|n|}$ (les η_i sont des paramètres formels)

9 déterminer les η_i en résolvant le système

$$\begin{cases} \lambda_k(\tilde{y})=\ell_k, & 1\leqslant k\leqslant r \\ b_{-s}(n)\,\tilde{y}_{n-s}+\cdots+b_s(n)\,\tilde{y}_{n+s}=0, & n\in[\![r, s-1]\!]\cup\boldsymbol{S} \end{cases} \tag{9.19}$$

ou échouer si ce n'est pas possible

10 renvoyer $\displaystyle\sum_{n=-d}^{d}\tilde{y}_n T_n(x)$ ♢

Remarque 9.18. Il est concevable qu'existent des équations différentielles auxquelles l'algorithme 9.16 est inapplicable, au sens où il échoue *pour tout N assez grand*. Cela se produit si l'espace engendré par les $s+k$ solutions test intersecte systématiquement le noyau (fixé, de dimension $s+k-r$) du membre gauche du système (9.18) avec dimension strictement plus grande que r.

Nous n'avons à ce jour ni exemple explicite de ce phénomène ni preuve qu'il n'arrive jamais. Cependant, lorsque les hypothèses **H1** et **H2** de la section suivante sont réalisées, la terminaison sans échec découle de la preuve de convergence. Une manière de remédier au problème en général serait de tirer au hasard les conditions initiales $(t_{i,n-s})_{n\in\boldsymbol{I}}$ qui définissent les solutions tests. En vue de la preuve de convergence, nous nous en tenons ici à la version déterministe où les conditions initiales sont fixées à l'identité. ♢

9.3.2 Convergence

Conservons les notations de l'algorithme 9.16. Soit y la fonction spécifiée par l'équation $L\cdot y=0$ et les conditions au bord données en entrée, et

$$y(x)=\sum_{n=-\infty}^{\infty} y_n T_n(x)$$

son développement de Tchebycheff. Nous démontrons maintenant que sous les hypothèses **H1** et **H2** ci-dessous, la sortie de l'algorithme converge rapidement vers $\pi_d(y)$, la série de Tchebycheff tronquée à l'ordre d de y. La preuve est en partie inspirée de l'analyse de l'algorithme de Miller généralisé [263, 256].

Lorsque ces hypothèses ne sont pas satisfaites, notre analyse ne dit rien sur la qualité des approximants qu'il est possible d'obtenir par l'algorithme 9.16. Celui-ci — s'il n'échoue pas — calcule tout de même une certaine approximation de $\pi_d(y)$,

qu'il demeure possible de valider en calculant une borne d'erreur par l'algorithme 9.32.

Supposons donc réalisées les conditions suivantes.

H1. Les racines complexes des équations caractéristiques d'une même arête du polygone de Newton de l'opérateur P (voir §9.2.4) sont simples et de modules deux à deux distincts.

H2. L'opérateur P n'a pas de singularité de queue aux indices $n \geqslant s$ — autrement dit, $\boldsymbol{S} = \varnothing$.

L'hypothèse **H1** permet d'appliquer la forme forte du théorème de Perron-Kreuser : d'après le théorème 3.25(b), soit

$$e_{-s}, ..., e_{-1}, e_1, ..., e_s$$

une base de germes à l'infini de solutions de la récurrence $P \cdot y = 0$ telles que

$$\forall i, \quad \frac{e_{i,n+1}}{e_{i,n}} \sim \alpha_i \, n^{\kappa_i} \tag{9.20}$$

où les α_i et κ_i sont donnés par le polygone de Newton comme rappelé en §9.2.4. Ces germes se prolongent en des suites solutions de la récurrence sur $\mathbb{N}$ (mais pas nécessairement, *a priori*, solutions symétriques ou même solutions sur $\mathbb{Z}$) d'après l'hypothèse **H2**. Cette dernière est là essentiellement pour alléger les notations. Elle entraîne que l'équation $L \cdot y = 0$ n'a pas de solutions polynomiales[8].

PROPOSITION 9.19. Soient

$$y_n^{(N)} = \tilde{y}_n, \qquad |n| \leqslant N,$$

les coefficients calculés par l'algorithme 9.16 (exécuté en arithmétique exacte), vus comme des fonctions du paramètre d'entrée N. Quand $N \to \infty$, l'erreur maximale sur un coefficient vérifie

$$\max_{n=-N}^{N} \left(y_n^{(N)} - y_n \right) = O(N^t \, e_{1,N})$$

pour un certain t indépendant de N. ♢

À d fixé et quand N tend vers l'infini, le polynôme calculé converge donc au moins exponentiellement vite vers la série de Tchebycheff tronquée à l'ordre d de y. La base de l'exponentielle est liée au comportement de la plus lentement décroissante des solutions convergentes de la récurrence.

Notre preuve fait appel au lemme suivant.

LEMME 9.20. Soient $(e_{0,n})_n$, ..., $(e_{s-1,n})_n$ des suites qui ne s'annulent pas pour n grand, chacune telle que

$$\frac{e_{i,n+1}}{e_{i,n}} \underset{n \to \infty}{\sim} \alpha_i \, n^{\kappa_i} \qquad (\kappa_i \in \mathbb{Q}, \alpha_i \in \mathbb{C} \setminus \{0\})$$

avec $\kappa_0 \leqslant \kappa_1 \leqslant \cdots \leqslant \kappa_{s-1}$ et $\kappa_i = \kappa_j \Rightarrow \alpha_i \neq \alpha_j$. Alors le déterminant de Casorati

$$C(n) = \begin{vmatrix} e_{0,n} & e_{1,n} & \cdots & e_{s-1,n} \\ e_{0,n+1} & & & e_{s-1,n+1} \\ \vdots & & & \vdots \\ e_{0,n+s-1} & e_{1,n+s-1} & \cdots & e_{s-1,n+s-1} \end{vmatrix}$$

vérifie

$$C(n) \sim e_{0,n} \, e_{1,n+1} \cdots e_{s-1,n+s-1} \prod_{\substack{i<j \\ \kappa_i \neq \kappa_j}} \left(\frac{\alpha_i}{\alpha_j} - 1 \right)$$

quand $n \to \infty$. ♢

8. Si un polynôme $y(x) = \sum_{n=-d}^{d} y_{|n|} \, T_n(x)$ de degré exactement d est solution de $L \cdot y = 0$, on a $(P \cdot y)_{d+s} = p_{-s}(d+s) \, y_d = 0$ et $y_d \neq 0$, donc $d \in \boldsymbol{S}$.

Démonstration. Posons $C(n) = e_{0,n}\, e_{1,n+1} \cdots e_{s-1,n+s-1}\, C'(n)$. On a alors[9]

$$C'(n) = \det\left(\frac{e_{j,n+i}}{e_{j,n+j}}\right)_{0\leqslant i,j<s} = \sum_{\sigma\in\mathfrak{S}_s} \varepsilon(\sigma) \prod_{j=0}^{s-1} \frac{e_{j,n+\sigma(j)}}{e_{j,n+j}}$$

où le terme d'indice σ a une croissance polynomiale en n d'exposant

$$\sum_{j=0}^{s-1} \big(\sigma(j) - j\big)\, \kappa_j.$$

Les termes dominants sont ceux qui maximisent

$$\sum_{j=0}^{s-1} \sigma(j)\, \kappa_j = \sum_{j=0}^{s-1} j\, \kappa_{\sigma(j)},$$

c'est-à-dire ceux pour lesquels les $\kappa_{\sigma(j)}$ sont en ordre croissant. On en déduit que lorsque n tend vers l'infini

$$\begin{aligned} C'(n) &= \sum_{\sigma\in\mathfrak{S}_s} \varepsilon(\sigma) \prod_{j=0}^{s-1} \mathbb{1}[\kappa_{\sigma(j)} = \kappa_j]\, \frac{e_{j,n+\sigma(j)}}{e_{j,n+j}} + o(1) \\ &= \det\left(\mathbb{1}[\kappa_i = \kappa_j]\, \frac{e_{j,n+i}}{e_{j,n+j}}\right)_{0\leqslant i,j<s} + o(1) \\ &= \prod_{\kappa\in\{\kappa_0,\dots,\kappa_s\}} \det\left(\frac{e_{j,n+i}}{e_{j,n+j}}\right)_{\substack{i,j:\\ \kappa_i=\kappa_j=\kappa}} + o(1). \end{aligned}$$

En multipliant pour tout j la colonne d'indice j et divisant la ligne de même indice par $n^{j\kappa}$ dans les déterminants du dernier produit, il vient

$$\det\left(\frac{e_{j,n+i}}{e_{j,n+j}}\right) = \det\left(n^{(j-i)\kappa}\, \frac{e_{j,n+i}}{e_{j,n+j}}\right) \underset{n\to\infty}{\to} \det\left(\alpha_j^{i-j}\right) = \prod_{\substack{i<j\\ \kappa_i=\kappa_j=\kappa}} \left(\frac{\alpha_i}{\alpha_j} - 1\right) \neq 0,$$

d'où le résultat. □

Démonstration de la proposition 9.19. Commençons par décrire la sortie de l'algorithme de sorte à pouvoir raisonner dessus aisément.

(a) La suite $(y_n^{(N)})_{n=-N}^{N}$ calculée s'étend en une solution $(y_n^{(N)})_{n\in\mathbb{Z}}$ de $P\cdot y^{(N)} = 0$, qui est caractérisée par les conditions

$$y_N^{(N)} = y_{N+1}^{(N)} = \cdots = y_{N+s-1}^{(N)} = 0$$

de l'étape 6 couplées au système (9.19) résolu à l'étape 9. En écrivant les formes linéaires $\lambda_1, \dots, \lambda_r \colon \mathcal{C} \to \mathbb{C}$ qui expriment les conditions au bord (9.2) sous la forme

$$\lambda_i(y) = \sum_{n=-\infty}^{\infty} \lambda_{i,n}\, y_n,$$

on en définit des « troncatures »

$$\lambda_i^{(N)}(y) = \sum_{n=-N}^{N} \lambda_{i,n}\, y_n$$

qui ont un sens sans supposer y convergente. En abusant un peu des notations, nous appliquerons les λ_i et $\lambda_i^{(N)}$ indifféremment à des fonctions, des séries de Tchebycheff formelles ou des suites interprétées comme des suites de coefficients

9. On convient par commodité de faire agir les permutations $\sigma \in \mathfrak{S}_s$ sur $[\![0, s-1]\!]$ plutôt que $[\![1, s]\!]$.

de Tchebycheff. On introduit de plus des formes linéaires

$$\lambda_{r+1} = \lambda_{r+1}^{(N)}, \ldots, \lambda_s = \lambda_s^{(N)}$$

de manière à écrire les $s - r$ dernières équations du système (les contraintes de symétrie des développements de Tchebycheff) sous la même forme que les r premières. Ainsi, le système (9.19) se réécrit

$$\lambda_i^{(N)}(y^{(N)}) = \sum_{n=-N}^{N} \lambda_{i,n}\, y_n^{(N)} = \ell_i, \quad 1 \leqslant i \leqslant s, \tag{9.21}$$

où l'on a aussi posé $\ell_{r+1} = \cdots = \ell_s = 0$.

(b) Soit maintenant

$$\Delta^{(N)} = \begin{vmatrix} e_{1,N} & \cdots & e_{s,N} & e_{-1,N} & \cdots & e_{-s,N} \\ \vdots & & \vdots & \vdots & & \vdots \\ e_{1,N+s-1} & \cdots & e_{s,N+s-1} & e_{-1,N+s-1} & \cdots & e_{-s,N+s-1} \\ \lambda_1^{(N)}(e_1) & \cdots & \lambda_1^{(N)}(e_s) & \lambda_1^{(N)}(e_{-1}) & \cdots & \lambda_1^{(N)}(e_{-s}) \\ \vdots & & \vdots & \vdots & & \vdots \\ \lambda_s^{(N)}(e_1) & \cdots & \lambda_s^{(N)}(e_s) & \lambda_s^{(N)}(e_{-1}) & \cdots & \lambda_s^{(N)}(e_{-s}) \end{vmatrix}, \tag{9.22}$$

et soit $\Delta_j^{(N)}$ le même déterminant où l'on a remplacé la colonne en e_j par

$${}^{\mathrm{t}}(0, \ldots, 0, \ell_1, \ldots, \ell_s).$$

D'après la règle de Cramer, sous réserve que $\Delta^{(N)} \neq 0$, la suite $(y_n^{(N)})_n$ se décompose sur la base $(e_j)_{j=-s}^{s}$ de $\ker P \subset \mathbb{C}^{\mathbb{Z}}$ sous la forme

$$y^{(N)} = \sum_{k=-s}^{s} \gamma_k^{(N)}\, e_k, \qquad \gamma_k^{(N)} = \frac{\Delta_k^{(N)}}{\Delta^{(N)}}. \tag{9.23}$$

On a $\Delta^{(N)} = 0$ si et seulement si l'algorithme échoue.

(c) De même, la « véritable » suite des coefficients de Tchebycheff de la fonction y est donnée par

$$y = \sum_{k=1}^{s} \gamma_k\, e_k, \qquad \gamma_k = \frac{\Delta_k}{\Delta}, \tag{9.24}$$

où Δ est le déterminant $\det(\lambda_i(e_j))_{1 \leqslant i,j \leqslant s}$, et Δ_j son analogue avec la j-ième colonne remplacée par ${}^{\mathrm{t}}(\ell_1, \ldots, \ell_s)$. Celui-ci est non nul par construction : l'orthogonalité d'un vecteur aux $s - r$ dernières lignes exprime qu'il s'agit des coordonnées sur $(e_1, \ldots, e_s)$ d'une solution de l'équation différentielle, et les r premières lignes donnent alors les conditions au bord correspondantes, or on a supposé les formes linéaires $\lambda_1, \ldots, \lambda_r$ indépendantes sur $\ker L$.

Notre objectif est maintenant de montrer que $\gamma_k^{(N)}$ converge « rapidement » vers γ_k quand N tend vers l'infini. Pour ce faire, nous étudions le comportement asymptotique des déterminants $\Delta^{(N)}$ et $\Delta_k^{(N)}$.

(d) Décomposons $\Delta^{(N)}$ en quatre blocs carrés comme suit :

$$\Delta^{(N)} = \begin{vmatrix} A & B \\ C & D \end{vmatrix},$$

(Toutes ces matrices sont des fonctions de N, mais nous abandonnons l'indice explicite pour des raisons de lisibilité.) Les blocs A, B et C sont inversibles pour N grand, les deux premiers d'après le lemme 9.20 ci-dessus, le troisième puisque $\det C \to \Delta \neq 0$ quand $N \to \infty$. La formule du complément de Schur implique

$$\Delta^{(N)} = -\det(B) \det(C) \det(I - C^{-1} D\, B^{-1} A).$$

Posons $\boldsymbol{e}_j = {}^{\mathrm{t}}(e_{j,N}, ..., e_{j,N+s-1})$. Le coefficient d'indice (i,j) de $B^{-1}A$ est

$$\begin{aligned}(B^{-1}A)_{i,j} &= \frac{\det(\boldsymbol{e}_{-1}, ..., \boldsymbol{e}_{-i+1}, \boldsymbol{e}_j, \boldsymbol{e}_{-i-1}, ..., \boldsymbol{e}_{-s})}{\det B} \\ &= \frac{(-1)^{i-1}\det(\boldsymbol{e}_j, \boldsymbol{e}_{-1}, ..., \widehat{\boldsymbol{e}_{-i}}, ..., \boldsymbol{e}_{-s})}{\det B}.\end{aligned}$$

On a $\kappa_j \leqslant \kappa_{-1} \leqslant \cdots \leqslant \kappa_{-s}$, donc le déterminant au numérateur et celui au dénominateur vérifient tous deux les hypothèses du lemme 9.20. Il vient

$$(B^{-1}A)_{i,j} = O\left(N^{\kappa_{-1}+\cdots+\kappa_{-i+1}-(i-1)\kappa_i}\,\frac{e_{j,N}}{e_{-i,N}}\right) = O\left(\frac{e_{j,N}}{e_{-i,N}}\right)$$

quand $N \to \infty$.

Les hypothèses (9.3) sur la forme des conditions au bord entraînent que $\lambda_{i,n} = O(n^r)$ quand $n \to \pm\infty$ pour $i \leqslant r$; et les suites $(\lambda_{i,n})_n$ pour $r+1 \leqslant i \leqslant s$ sont nulles à partir d'un certain rang. Les coefficients de D satisfont donc

$$D_{i,j} = \lambda_i^{(N)}(e_{-j}) = O(N^r e_{-j,N}).$$

On en tire l'estimation

$$(D\,B^{-1}A)_{i,j} = O(N^r e_{j,N})$$

pour la j-ième colonne de $D\,B^{-1}A$. Comme

$$C_{i,j} = \lambda_i^{(N)}(e_j) = \lambda_i(e_j) + O(N^r e_{j,N}),$$

on obtient aussi

$$(C^{-1}\,D\,B^{-1}A)_{i,j} = O(N^r e_{j,N})$$

d'où finalement

$$\begin{aligned}\Delta^{(N)} &= -\det(B)\det(C)\,(1 - \operatorname{tr}(C^{-1}\,D\,B^{-1}A) + O(\|C^{-1}\,D\,B^{-1}A\|^2)) \\ &= -\det(B)\,(\Delta + O(N^r e_{1,N})).\end{aligned}$$

(e) Passons aux déterminants modifiés $\Delta_k^{(N)}$. On pose comme dans le cas prédécent

$$\Delta_k^{(N)} = \begin{vmatrix} A_k & B_k \\ C_k & D_k \end{vmatrix}.$$

Pour $k > 0$, le même raisonnement qu'au point (d) (à ceci près que la matrice C_k peut être singulière) conduit à l'estimation

$$\begin{aligned}\Delta_k^{(N)} &= -\det(B)\det(C_k - D\,B^{-1}A_k) \\ &= -\det(B)\,(\det(C_k) + O(N^r e_{1,N})) \\ &= -\det(B)\,(\Delta_k + O(N^r e_{1,N})),\end{aligned}$$

d'où

$$\gamma_k^{(N)} = \frac{\Delta_k^{(N)}}{\Delta^{(N)}} = \gamma_k + O(N^r e_{1,N}), \qquad k > 0.$$

(f) Dans le cas $k < 0$, écrivons

$$\Delta_k^{(N)} = -\det(C)\det(B_k - A\,C^{-1}\,D_k).$$

Les bornes coefficient par coefficient immédiates sur A et D entraînent

$$(C^{-1}\,D_k)_{i,j} = O(N^r e_{-j,N+s-1}),$$

puis

$$(A\,C^{-1}\,D_k)_{i,j} = O(N^r e_{1,N}\,e_{-j,N+s-1}) = o(e_{-j,N}),$$

de sorte que l'on a

$$(B_k + A\,C^{-1}\,D_k)_{i,j} \sim e_{-j,N+i-1}, \qquad j \neq -k.$$

Pour $j=-k$ cependant, la j-ième colonne de B_k est nulle et celle de D_k est constante, donc

$$(B_k + A\,C^{-1}\,D_k)_{i,j} = O(e_{1,N}), \qquad j=-k.$$

Il s'ensuit que

$$\begin{aligned}\det(B_k + A\,C^{-1}\,D_k) &= O(e_{-1,N+s-1}\cdots \widehat{e_{k,N+s-1}}\cdots e_{-s,N+s-1}e_{1,N}) \\ &= O\left(\frac{N^\tau \det(B)}{e_{k,N}}\,e_{1,N}\right)\end{aligned}$$

où $\tau \leqslant \sum_{j\neq k}(s-j)\,\kappa_{-j}$, d'où

$$\gamma_k^{(N)} = \frac{\Delta_k^{(N)}}{\Delta^{(N)}} = \frac{-\det(B)\det(C)\,N^\tau}{-\det(B)\det(C)\,(1+O(e_{1,N}))}\,\frac{e_{1,N}}{e_{k,N}}$$

et finalement

$$\gamma_k^{(N)} = O\left(N^\tau\,\frac{e_{1,N}}{e_{k,N}}\right), \qquad k<0.$$

(g) En combinant les décompositions (9.23) et (9.24) avec les conclusions des points (e) et (f), il vient finalement

$$y_n^{(N)} = y_n + O\left(N^{\max(r,\tau)}\,e_{1,N}\sum_{k=1}^{s}\left(e_{k,n}+\frac{e_{-k,n}}{e_{-k,N}}\right)\right)$$

quand $N\to\infty$, uniformément en n. □

9.3.3 Retour sur la méthode tau de Lánczos

Une seconde méthode populaire pour le calcul approché de développements de Tchebycheff est la méthode τ de Lánczos [153, 154]. Il a déjà été observé, par Fox [97] puis de manière plus générale (et dans un langage différent) par El Daou, Ortiz et Samara [84] que les méthodes de Clenshaw et de Lánczos sont en fait équivalentes, au sens où l'on peut les exprimer dans un même cadre et les ajuster de telle façon qu'elles produisent des approximations identiques. Comment le recours à la récurrence de Tchebycheff se place-t-il de ce point de vue ?

L'examen de cette question jette une lumière un peu différente sur l'algorithme 9.16 et montre comment utiliser la récurrence de Tchebycheff dans l'application de la méthode τ. En passant, nous observerons que l'expression *sur la base monomiale* du polynôme $\tilde{y}$ renvoyé par l'algorithme 9.16 est elle-même calculable en temps linéaire en N. À titre de comparaison, la meilleure complexité arithmétique connue pour la conversion d'un polynôme arbitraire de degré d de la base de Tchebycheff à la base monomiale (ou inversement) est $\Theta(M(d))$ [197, 35, 18], où $M(\cdot)$ représente le coût de la multiplication de polynômes (voir §7.2).

On considère toujours une équation différentielle $L\cdot y=0$, d'ordre r, d'une solution de laquelle on recherche une approximation polynomiale de degré d. Supposons pour simplifier que l'équation n'a pas de solutions polynomiales.

En quelques mots, la méthode τ procède comme suit. La première étape consiste à calculer $L\cdot p$ pour un polynôme p de degré d à coefficients indéterminés. Comme $(\ker L)\cap\mathbb{C}[x]=\{0\}$, le résultat est de degré supérieur à d. On introduit des inconnues additionnelles $\tau_d,\ldots,\tau_{d+m}$ en nombre choisi de sorte que le système linéaire

$$\begin{cases} L\cdot p = \tau_d T_d + \cdots + \tau_{d+m} T_{d+m} \\ \lambda_i(p) = \ell_i \qquad\qquad (1\leqslant i\leqslant r) \end{cases} \tag{9.25}$$

ait une solution (unique, de préférence). La sortie de l'algorithme est la valeur de p obtenue en résolvant ce système : il s'agit d'une solution exacte de « l'équation projetée » $\pi_{d-1}(L\cdot y)=0$.

Notons $p=\sum_n p_n T_n$, et étendons la suite (τ_n) en posant

$$\tau_n = 0 \quad \text{pour} \quad n\in\mathbb{N}\setminus[\![d+1, d+m]\!], \qquad \tau_{-n}=\tau_n.$$

Le système (9.25) entraîne que $P\cdot(p_n)=\frac{1}{2}\,Q\cdot(\tau_n)$ où P et Q sont les opérateurs du théorème 9.3. En notant $\operatorname{Supp} u=\{|n|:u_n\neq 0\}$, on voit aussi de par la forme explicite de Q que $\operatorname{Supp}(Q\cdot\tau)\subset[\![d,d+m+1]\!]$.

Fait 9.21. Les coefficients p_n du résultat de la méthode τ sont donc calculables par la récurrence de Tchebycheff, à partir d'un petit nombre de conditions initiales données au voisinage de l'indice $|n|=d$. ♢

Inversement, considérons le polynôme $\tilde{y}$ calculé par l'algorithme 9.16, et soit

$$v=\sum_n v_n\,T_n=L\cdot\tilde{y}.$$

On a $P\cdot\tilde{y}=Q\cdot v$ d'après le théorème 9.3. La définition de $\tilde{y}$ dans l'algorithme implique aussi que $(P\cdot\tilde{y})_n=0$ pour $|n|\leqslant N-s$ (puisque les $\tilde{y}_n$, $|n|\leqslant N$ sont des combinaisons linéaires de suites $(t_{i,n})_{|n|\leqslant N}$ obtenues en déroulant la récurrence $P\cdot t_i=0$) et pour $|n|>N+s$ (puisque $\tilde{y}_n=0$ dans ce cas), de sorte que

$$\operatorname{Supp}(Q\cdot v)\subset[\![N-s,N+s-1]\!].$$

Or, on peut vérifier que l'opérateur P associé à $L=(\frac{\mathrm{d}}{\mathrm{d}x})^r$ par l'algorithme de Paszkowski est $P=\delta_r(n)$ — en effet, dans le formalisme de Benoit et Salvy [21], il doit satisfaire $Q^{-1}P=I^{-r}$. La relation $\delta_r(n)\cdot u=Q\cdot v$ est donc équivalente à $u^{(r)}=v$, et

$$v(x)=\frac{\mathrm{d}^r}{\mathrm{d}\,x^r}\sum_{|n|>r}\frac{(P\cdot\tilde{y})_n}{\delta_r(n)}\,T_n(x)=\sum_{N-s\leqslant|n|<N+s}\frac{(P\cdot\tilde{y})_n}{\delta_r(n)}\,T_n^{(r)}(x)\tag{9.26}$$

Ainsi, la sortie $\tilde{y}(x)$ de l'algorithme 9.16 est solution d'une équation différentielle inhomogène de la forme

$$L\cdot\tilde{y}=\tau_{N-s}\,T_{N-s}^{(r)}(x)+\cdots+\tau_{N+s-1}\,T_{N+s-1}^{(r)}(x).$$

(Cependant, le support de la suite (v_n) n'est en général pas creux.)

Proposition 9.22. L'expression sur la base monomiale du polynôme $\tilde{y}(x)$ renvoyé par l'algorithme 9.16 est calculable en $O(N)$ opérations arithmétiques quand $N\to\infty$, tous les autres paramètres étant fixés. ♢

Démonstration. Le développement de Taylor d'une solution d'une équation différentielle inhomogène $L\cdot u=v$ satisfait une récurrence avec au membre gauche un opérateur de récurrence qui ne dépend que de L, et au membre droit la suite des coefficients de v. Or on a $L\cdot\tilde{y}=v$, où la fonction v est donnée par (9.26). Les coefficients $(P\cdot\tilde{y})_n/\delta_r(n)$ qui y apparaissent s'obtiennent en temps constant (vis-à-vis de N) à partir des quelques derniers coefficients de Tchebycheff de $\tilde{y}$. On en déduit v_0, ..., v_{N+s-r} en $O(N)$ opérations en appliquant de manière répétée la récurrence inhomogène

$$T'_{n-1}(x)=-T'_{n+1}(x)+2\,x\,T'_n(x)+2\,T_n(x),$$

qui provient de la dérivation de (9.9) ; puis ceux du développement sur la base monomiale de $\tilde{y}$, *via* la relation de récurrence inhomogène déjà mentionnée. □

9.4 Développements de Tchebycheff des fractions rationnelles

9.4.1 Introduction

Nous savons à ce stade calculer approximativement le développement de Tchebycheff d'une fonction D-finie, sans contrôle autre qu'asymptotique sur l'erreur commise.

Avant de voir comment valider après coup la qualité des polynômes d'approximation construits de cette façon, nous faisons dans cette section un détour par le cas particulier des fractions rationnelles. Cette organisation rappelle celle du calcul de séries majorantes au chapitre 5, où nous commencions par calculer des séries majorantes des coefficients d'une équation différentielle pour en déduire ensuite des séries majorantes des solutions. Ici, être en mesure de développer en série de Tchebycheff, de façon garantie, les fractions rationnelles qui apparaissent comme coefficients des équations nous sera utile lors de l'étape de validation.

Les questions abordées sont peu ou prou les mêmes que dans le cas général des fonctions D-finies. On cherche à obtenir une relation de récurrence sur les coefficients y_n d'une fraction rationnelle y ; à s'en servir pour calculer approximativement les y_n ; et à certifier la qualité des approximations polynomiales ainsi construites.

Notre outil principal est le changement de variable $x = \frac{1}{2}(z + z^{-1})$ suivi d'une décomposition en éléments simples. Des idées analogues ont été mises en œuvre par le passé à des fins proches des nôtres, par exemple pour déterminer les coefficients y_n en forme close [83, 170]. Il s'avère en effet que la suite $(y_n)_{n\in\mathbb{N}}$ vérifie une récurrence linéaire à coefficients *constants* : trouver cette récurrence ou exprimer les y_n en forme close revient donc essentiellement au même. Cependant, nous aurons besoin par la suite de quelques résultats sur le coût des algorithmes que nous n'avons pas trouvés dans la littérature. Notre souci de ce point de vue est d'éviter les conversions de la base de Tchebycheff vers la base monomiale ou inversement, afin d'aboutir à une complexité arithmétique linéaire.

9.4.2 Récurrence et expression explicite

Soit $y(x) = a(x)/b(x) \in \mathbb{Q}[x]$ une fraction rationnelle sans pôle sur $[-1;1]$. On note $(y_n)_{n\in\mathbb{Z}}$, $(a_n)_{n\in\mathbb{Z}}$ et $(b_n)_{n\in\mathbb{Z}}$ les suites (symétriques) respectives des coefficients de Tchebycheff de y, a et b.

PROPOSITION 9.23. La suite $(y_n)_{n\in\mathbb{Z}}$ vérifie la relation de récurrence linéaire à coefficients constants

$$b\left(\frac{S+S^{-1}}{2}\right)\cdot(y_n)_{n\in\mathbb{Z}} = (a_n)_{n\in\mathbb{Z}}. \qquad \diamondsuit$$

DÉMONSTRATION. Ce n'est à vrai dire que le cas limite $r = 0$ du théorème 9.3, mais une preuve directe est facile : on écrit simplement

$$\sum_{i=-\deg b}^{\deg b} b_i\, z^i \sum_{n=-\infty}^{\infty} y_n\, z^n = \sum_{n=-\infty}^{\infty}\left(\sum_{i=-\deg b}^{\deg b} b_i\, y_{n-i}\right) z^n = \sum_{n=-\infty}^{\infty} a_n\, z^n, \quad x = \frac{z+z^{-1}}{2},$$

et l'on identifie les coefficients des puissances de z. □

Cette récurrence souffre des mêmes problèmes liés à l'existence de solutions divergentes et à l'accès aux conditions initiales que celle du théorème 9.3. Mais il est aisé ici d'isoler le sous-espace des solutions convergentes : on sépare les puissances positives de z des puissances négatives dans le développement de Laurent

$$\hat{y}(z) = y\left(\frac{z+z^{-1}}{2}\right) = \sum_{n=-\infty}^{\infty} y_n\, z^n, \qquad \rho^{-1} < |z| < \rho, \tag{9.27}$$

par décomposition en éléments simples. Du point de vue du calcul, mieux vaut prendre comme point de départ la factorisation sans carré du dénominateur de $\hat{y}$,

$$\beta(z) = z^{\deg b}\, b\left(\frac{z+z^{-1}}{2}\right) = \beta_1(z)\,\beta_2(z)^2 \cdots \beta_k(z)^k, \tag{9.28}$$

et écrire la décomposition en éléments simples sur les complexes de $\hat{y}(z)$ sous la forme

$$\hat{y}(z) = q(z) + \sum_{i=1}^{k} \sum_{\beta_i(\zeta)=0} \sum_{j=1}^{i} \frac{h_{i,j}(\zeta)}{(\zeta - z)^j}, \tag{9.29}$$

où

$$q(z) = \sum_n q_n z^n \in \mathbb{Q}[z], \quad h_{i,j} \in \mathbb{Q}[z].$$

Les $h_{i,j}$ se calculent efficacement par l'algorithme de Bronstein-Salvy [43] (cf. §5.3.2).

De (9.29) découle une identité de la forme (9.27) en développant en série à l'origine les éléments simples associés aux pôles ζ de module $|\zeta| > 1$, et à l'infini ceux avec $|\zeta| < 1$. Ces développements convergent respectivement pour $|z| < |\zeta|$ et $|z| > |\zeta^{-1}|$, donc la série de Laurent bi-infinie obtenue converge sur la couronne $\rho^{-1} < |\zeta| < \rho$. Le développement à l'infini de

$$\frac{h_{i,j}(\zeta)}{(\zeta - z)^j} = \frac{(-1)^j z^{-j} h_{i,j}(\zeta)}{(1 - \zeta z^{-1})^j}$$

ne contribue pas aux coefficients de z^n d'exposant positif. Par unicité du développement de Laurent de $\hat{y}$ au voisinage du cercle unité, on en déduit[10]

$$\sum_{n=0}^{\infty} y_n z^n = q(z) + \sum_{i=1}^{k} \sum_{\substack{\beta_i(\zeta)=0 \\ |\zeta|>1}} \sum_{j=1}^{i} \frac{h_{i,j}(\zeta)}{(\zeta - z)^j}. \tag{9.30}$$

On extrait le coefficient de z^n dans (9.30), et l'on invoque la symétrie de la suite (y_n), afin d'en tirer une expression explicite de y_n en termes des racines de $b\big(\frac{1}{2}(z + z^{-1})\big)$.

Proposition 9.24. Les coefficients du développement de Tchebycheff

$$y(x) = \sum_{n=-\infty}^{\infty} y_{|n|} T_n(x)$$

sont donnés par la formule

$$y_n = q_n + \sum_{i=1}^{k} \sum_{j=1}^{i} \sum_{\substack{\beta_i(\zeta)=0 \\ |\zeta|>1}} \binom{n+j-1}{j-1} h_{i,j}(\zeta)\, \zeta^{-n-j} \qquad (n \geqslant 0) \tag{9.31}$$

où les $q_n \in \mathbb{Q}$, $\beta_i \in \mathbb{Q}[z]$ et $h_{i,j} \in \mathbb{Q}(z)$ sont définis par les équations (9.28) et (9.29). $\diamondsuit$

Observons que l'équation (9.30) fournit implicitement une récurrence d'ordre $\deg b$ sur la suite $(y_n)_{n\in\mathbb{N}}$, alors que celle qui résulte de la proposition 9.23 a un ordre double. En revanche, la nouvelle récurrence est en général à coefficients algébriques et non rationnels. Il est naturel de se demander si une construction analogue est possible dans le cas général des fonctions D-finies. Nous n'avons hélas pas encore pu consacrer à cette question le temps qu'elle mériterait.

9.4.3 Borne sur le reste

Une fois la formule établie, nous pouvons facilement borner l'erreur commise en tronquant la série de Tchebycheff de y.

10. Peut-être vaut-il la peine de relever pour éviter toute confusion que dans l'expression

$$\hat{y}(z) = q(z) + \sum_{i=1}^{k} \sum_{\substack{\beta_i(\zeta)=0 \\ |\zeta|>1}} \sum_{j=1}^{i} \left(\frac{h_{i,j}(\zeta)}{(\zeta - z)^j} + \frac{h_{i,j}(\zeta^{-1})}{(\zeta^{-1} - z)^j} \right),$$

le développement de Laurent d'un terme isolé n'est *pas symétrique* pour $j > 1$.

PROPOSITION 9.25. Soit $y \in \mathbb{Q}(x)$ une fraction rationnelle sans pôle sur le disque elliptique E_ρ (voir équation (9.7) page 177). En reprenant toutes les notations des équations (9.27) à (9.29), on a

$$\left\| \sum_{n>d} y_n T_n \right\|_\infty \leqslant \sum_{i=1}^{k} \sum_{j=1}^{i} \sum_{\substack{\beta_i(\zeta)=0 \\ |\zeta|>1}} \frac{|h_{i,j}(\zeta)|\,(d+2)^{j-1}}{(|\zeta|-1)^j}\,|\zeta|^{-d-1}$$

pour tout ordre de troncature $d \geqslant \deg q$. La borne est $O(d^{\deg b}\,\rho^{-d})$ quand $d \to \infty$. ♢

DÉMONSTRATION. Tout d'abord, on a bien entendu

$$\left\| \sum_{n>d} y_n T_n \right\|_\infty \leqslant \sum_{n>d} |y_n|$$

puisque $\|T_n\|_\infty \leqslant 1$ pour tout n. À l'aide de l'inégalité valable pour $t<1$

$$\sum_{n>d} \binom{n+j-1}{j-1} t^{n+j} \leqslant (d+2)^{j-1}\,t^{d+1} \sum_{n=0}^{\infty} \binom{n+j-1}{j-1} t^{n+j} = \frac{(d+2)^{j-1}\,t^{d+j+1}}{(1-t)^j},$$

on tire de l'expression explicite établie à la proposition 9.24 la borne

$$\begin{aligned} \sum_{n>d} |y_n| &\leqslant \sum_{n>d} \sum_{i=1}^{k} \sum_{j=1}^{i} \sum_{\substack{\beta_i(\zeta)=0 \\ |\zeta|>1}} \binom{n+j-1}{j-1} |h_{i,j}(\zeta)|\,|\zeta|^{-n-j} \\ &\leqslant \sum_{i=1}^{k} \sum_{j=1}^{i} \sum_{\substack{\beta_i(\zeta)=0 \\ |\zeta|>1}} \frac{|h_{i,j}(\zeta)|\,(d+2)^{j-1}}{(|\zeta|-1)^j}\,|\zeta|^{-d-1}. \end{aligned}$$

L'estimation asymptotique provient de ce que $|\zeta|>1$ entraîne en fait $|\zeta|>\rho$ lorsque ζ vérifie l'équation $b\big(\frac{1}{2}\,(\zeta+\zeta^{-1})\big)=0$. □

9.4.4 Calcul

Reste à s'assurer que les résultats précédents induisent effectivement un algorithme de complexité arithmétique linéaire.

LEMME 9.26. Le calcul du produit $a\,b$, où les opérandes $a, b \in \mathbb{Q}[x]$ et le résultat sont tous représentés sur la base de Tchebycheff, demande $O((\deg a)\,(\deg b))$ opérations. ♢

DÉMONSTRATION. On a pour tous i et j

$$2\,T_i\,T_j = T_{i+j} + T_{i-j}. \tag{9.32}$$

Il suffit pour chaque couple $(i,\,j)$ de mettre à jour les coefficients de $T_{|i\pm j|}$ dans $a\,b$ en fonction de ceux de T_i dans a et de T_j dans b. □

À diviseur fixé, l'algorithme classique de division euclidienne de polynômes [249, Algorithm 2.5] s'exécute en temps linéaire vis-à-vis du degré du dividende. Entrée et sortie y sont habituellement représentées sur la base monomiale, mais l'algorithme s'adapte sans difficulté à d'autres bases.

LEMME 9.27. La division euclidienne $a = b\,q + r$ $(\deg r < \deg b)$ avec $a, b, q, r \in \mathbb{Q}[x]$ représentés sur la base de Tchebycheff demande $O(\deg a)$ opérations à b fixé. ♢

DÉMONSTRATION. Supposons $n = \deg a > \deg b = m$. L'algorithme classique de division euclidienne s'appuie principalement sur le fait que

$$\deg\,(a - b_m^{-1}\,a_n\,x^{n-m}\,b) < n$$

où $a = \sum_i a_i\, x^i$ et $b = \sum_i b_i\, x^i$. De la formule de multiplication (9.32) découle l'inégalité analogue

$$\deg\left(a - 2\, b_m^{-1}\, a_n\, T_{n-m}\, b\right) < n$$

où a_k, b_k sont maintenant les coefficients de a et b sur la base de Tchebycheff. Pour faire l'ensemble du calcul sur cette base, il suffit de remplacer les affectations répétées de la forme

$$a \leftarrow a - b_m^{-1}\, a_n\, x^{n-m}\, b$$

de l'algorithme classique par

$$a \leftarrow a - 2\, b_m^{-1}\, a_n\, T_{n-m}\, b.$$

Le polynôme $T_{n-m}\, b$ ayant au plus $2\, m$ coefficients non nuls, chacune de ces étapes prend un temps constant vis-à-vis de n. L'algorithme en effectue au plus $n - m$, pour un total de $O(n)$ opérations arithmétiques. □

On aboutit en résumé à l'algorithme 9.26. En vue de la suite, il prend en entrée à la fois une fraction rationnelle dont les degrés du numérateur et du dénominateur sont supposés bornés, et un second facteur au numérateur, de degré de l'ordre de celui attendu en sortie, mais déjà écrit sur la base de Tchebycheff.

Algorithme 9.28. ChebExpandRatpoly(y, f, ε)

Entrée. Une fraction rationnelle $y(x) = a(x)/b(x) \in \mathbb{Q}(x)$. Les coefficients de Tchebycheff d'un polynôme $f(x) = \sum_{n=-d}^{d} f_n\, T_n(x) \in \mathbb{Q}[x]$. Une précision cible $\varepsilon > 0$.

Sortie. Les coefficients de Tchebycheff d'un polynôme $\tilde{y}(x)$ tel que $\left\| \tilde{y} - \frac{a}{b}\, f \right\| \leqslant \varepsilon$.

1 développer a et b sur la base de Tchebycheff

2 calculer $g := a\, f$, en opérant sur la base de Tchebycheff

3 calculer le quotient q et le reste r de la division euclidienne de g par b

4 calculer la décomposition en éléments simples de $\hat{w}(z) = w(x) = \dfrac{r(x)}{b(x)}$, où $x = \dfrac{z + z^{-1}}{2}$, par l'algorithme de Bronstein-Salvy, et en déduire celle (9.29) de $\hat{y}(z) = \dfrac{a(x)}{b(x)}\, f(x) = q(x) + w(x)$

5 déterminer $d' \geqslant \deg q$ tel que $\left\| \displaystyle\sum_{n > d'} y_n\, T_n \right\|_\infty \leqslant \dfrac{\varepsilon}{4}$ par la proposition 9.25

6 calculer ρ_- et ρ_+ tels que $\beta(\zeta) = 0 \wedge |\zeta| > 1 \Rightarrow 1 < \rho_- \leqslant |\zeta| \leqslant \rho_+$

7 calculer $M \geqslant \displaystyle\sum_{i=1}^{k} \sum_{j=1}^{i} j\, (\deg \beta_i) \sup_{\rho_- \leqslant |\zeta| \leqslant \rho_+} \left(|h'_{i,j}(\zeta)| + |\zeta^{-1}\, h_{i,j}(\zeta)| \right) \rho_-^{-j}$

8 $\varepsilon' := \min\left(\rho_- - 1, M^{-1}\, (1 - \rho_-^{-1})^{D+1}\, \dfrac{\varepsilon}{4} \right)$ où $D = \deg b$

9 calculer des approximations $\tilde{\zeta} \in \mathbb{Q}[i]$ des racines ζ de β, à précision $|\tilde{\zeta} - \zeta| < \varepsilon'$

10 pour $0 \leqslant n \leqslant d'$

11 $\tilde{y}_n := q_n + \mathrm{Re}\left(\displaystyle\sum_{i=1}^{k} \sum_{j=1}^{i} \sum_{\beta_i(\zeta) = 0 \wedge |\zeta| > 1} \binom{n+j-1}{j-1}\, h_{i,j}(\tilde{\zeta})\, \tilde{\zeta}^{-n-j} \right)$

12 renvoyer $\tilde{y}(x) = \displaystyle\sum_{n=-d'}^{d'} \tilde{y}_n\, T_n(x)$ ♢

Les détails de l'algorithme tel qu'il est présenté ne servent guère qu'à établir l'estimation de complexité. En pratique, toutes sortes de variantes et améliorations sont possibles. Par exemple, à l'étape 7, on peut remplacer les bornes uniformes sur

la couronne par des estimations au voisinage de chaque racine, faciles à obtenir en arithmétique d'intervalles complexes. À l'étape 11, mieux vaut regrouper deux à deux les racines complexes conjuguées de β. Les valeurs successives de $\tilde{y}_n$ se calculent naturellement de façon itérative. On peut aussi calculer les $h_{i,j}(\zeta)\ \zeta^{-n-j} \in \mathbb{Q}[\zeta]$ symboliquement par multiplications par ζ modulo $\beta_i(\zeta)$ répétées, et ne les remplacer par des approximations numériques qu'au dernier moment.

PROPOSITION 9.29. L'algorithme 9.26 est correct. Il s'exécute en $O(d + \log(\varepsilon^{-1}))$ opérations arithmétiques, tous les autres paramètres étant fixés. ♢

DÉMONSTRATION. Montrons d'abord la borne d'erreur $\|\tilde{y} - y\|_\infty \leqslant \varepsilon$. Soient

$$M_0 = \sup_{\rho_- \leqslant |\zeta| \leqslant \rho} |h'_{i,j}(\zeta)|, \qquad M_1 = \sup_{\rho_- \leqslant |\zeta| \leqslant \rho} |\zeta^{-1} h_{i,j}(\zeta)|.$$

La dérivée par rapport à ζ de $h_{i,j}(\zeta)\ \zeta^{-n-j}$ est bornée sur $A = \{\zeta : \rho_- \leqslant |\zeta| \leqslant \rho_+\}$ par

$$\big(M_0 + (n+j)\, M_1\big)\, |\zeta|^{-n-j}.$$

La condition $|\tilde{\zeta} - \zeta| < \rho_- - 1$ issue de l'étape 8 entraîne que $[\zeta, \tilde{\zeta}] \subset A$ pour chacun des ζ considérés. En comparant la formule de l'étape 11 à (9.31), on obtient

$$\begin{aligned}
|\tilde{y}_n - y_n| &\leqslant \sum_{i=1}^{k} \sum_{j=1}^{i} \sum_{\substack{\beta_i(\zeta)=0 \\ |\zeta|>1}} \binom{n+j-1}{j-1} \Big| h_{i,j}(\tilde{\zeta})\, \tilde{\zeta}^{-n-j} - h_{i,j}(\zeta)\, \zeta^{-n-j} \Big| \\
&\leqslant \sum_{i=1}^{k} \sum_{j=1}^{i} \sum_{\substack{\beta_i(\zeta)=0 \\ |\zeta|>1}} \binom{n+j-1}{j-1} \big(M_0 + (n+j)\, M_1\big)\, |\zeta|^{-n-j}\, |\tilde{\zeta} - \zeta| \\
&\leqslant \sum_{i=1}^{k} \sum_{j=1}^{i} j\, (\deg \beta_i) \binom{n+j}{j} \big(M_0 + M_1\big)\, \rho_-^{-n-j}\, \varepsilon' \\
&\leqslant M \binom{n+D}{D} \rho_-^{-n}\, \varepsilon'.
\end{aligned}$$

Par conséquent

$$\|\tilde{y} - y\|_\infty \leqslant \sum_{n=-d'}^{d'} |\tilde{y}_n - y_n| + 2 \left\| \sum_{n>d'} y_n T_n \right\|_\infty \leqslant \frac{2\, M \varepsilon'}{(1 - \rho_-^{-1})^{D+1}} + 2\, \frac{\varepsilon}{4} \leqslant \varepsilon.$$

Passons à l'analyse de complexité. Les étapes 1 et 4 à 8 de même que chaque itération de la boucle finale (en retenant les valeurs de $\tilde{\zeta}^{-n}$ d'une itération sur l'autre) ont un coût constant. Les étapes 2 et 3 prennent un temps linéaire en d, d'après les lemmes 9.26 et 9.27, et ne dépendent pas de ε. Enfin, il est connu que l'on peut calculer des approximations à précision absolue η des racines d'un polynôme à coefficients entiers en $O(\eta^{-1})$ opérations arithmétiques [196, Theorem 1.1(d)]. (Notons en passant que la complexité *binaire* de l'algorithme justifiant cette assertion est elle-même quasi-linéaire en ε^{-1}. Les dépendances en le degré du polynôme de ces deux complexités sont polynomiales de petit degrés.) Or $\varepsilon' = \Omega(\varepsilon)$ puisque M ne dépend ni de ε ni de d, donc le coût de l'étape 9 est $O(\varepsilon^{-1})$. □

9.5 Validation

9.5.1 Situation

Dans cette section, nous supposons calculé un polynôme de degré d

$$p(x) = \tilde{y}(x) = \sum_{n=-d}^{d} p_n\, T_n(x) \tag{9.33}$$

présumé fournir une bonne approximation sur $[-1;1]$ d'une fonction y spécifiée par une équation différentielle. Nous nous limitons au cas où l'équation est accompagnée de conditions initiales à l'origine[11]. Comme énoncé en introduction, nous cherchons alors à déterminer une borne R raisonnablement fine sur $\|y-p\|_\infty$.

L'idée principale est de vérifier par un calcul en arithmétique d'intervalles que p est « presque solution » d'une équation de point fixe (contractante !) équivalente à l'équation différentielle qui définit y (cf. [179, 72]). L'algorithme auquel nous avons recours peut se voir comme une version effective de la preuve du théorème de Cauchy-Lipschitz pour les équations différentielles linéaires. Il s'agit ainsi d'une instance très simple des méthodes d'inclusion fonctionnelle déjà présentées en §6.1.2. Ce genre de stratégie apparaît habituellement dans le cadre de calculs par intervalles sur des développements de Taylor en une ou plusieurs variables (modèles de Taylor et variantes [164, 192]). Une exception notable est fournie par les travaux de Kaucher, Miranker et d'autres [86, 139, 140] mentionnés en début de chapitre, qui couvrent le cas des séries de Tchebycheff.

Quant aux détails de l'algorithme, ils sont motivés en premier lieu par le souci d'aboutir à une complexité arithmétique linéaire. C'est le sens, notamment, du recours aux sous-routines mises en place dans la section précédente pour développer les fractions rationnelles.

9.5.2 Idée

Avant d'énoncer l'algorithme de validation, voyons rapidement à quoi il ressemble dans le cas d'une équation différentielle d'ordre 1. Considérons l'équation

$$y'(x) = a(x)\,y(x), \qquad y(0) = c, \tag{9.34}$$

où a est une fraction rationnelle sans pôle sur $[-1;1]$. En introduisant l'opérateur

$$\tau\colon \quad y \mapsto \left(x \mapsto c + \int_0^x a(t)\,y(t)\,\mathrm{d}t \right), \tag{9.35}$$

le problème (9.34) se met sous la forme de point fixe $\tau(y) = y$. Soit y l'unique solution.

Supposons un instant disposer, en plus du polynôme (9.33), d'une borne d'erreur candidate R. Comment vérifier par le calcul que $\|p-y\|_\infty \leqslant R$? Plaçons-nous dans l'espace de Banach des fonctions analytiques sur un ouvert $\Omega \subset \mathbb{C}$ contenant $[-1;1]$, munies de la norme de la convergence uniforme, et notons

$$\mathcal{B}(g,R) = \{f : \|f-g\|_\infty \leqslant R\} \tag{9.36}$$

la boule fermée centrée en g et de rayon R. C'est un sous-espace métrique complet. On vérifie de plus que

$$\|\tau^j(f) - \tau^j(g)\|_\infty \leqslant \gamma_j\,\|f-g\|_\infty, \qquad \text{où } \gamma_j = \frac{\|a\|_\infty^j}{j!}. \tag{9.37}$$

Supposons la boule $\mathcal{B}(p,R)$ stable par τ^i pour un certain i. Comme $\gamma_j \to 0$ quand $j \to \infty$, la restriction de τ^i à $\mathcal{B}(p,R)$ possède un itéré contractant, et donc admet un unique point fixe. La solution y appartient alors à $\mathcal{B}(p,R)$.

Concrètement, on calcule une approximation p_i de $\tau^i(p)$ ainsi qu'une borne R_i sur l'erreur de *ce calcul* — par exemple, à l'aide d'une implémentation de τ en arithmétique d'intervalles — de manière à ce que $\|f-p\|_\infty \leqslant R \Rightarrow \|\tau^i(f) - p_i\| \leqslant R_i$. On cherche à vérifier que

$$\tau^i\big(\mathcal{B}(p,R)\big) \subset \mathcal{B}(p_i,R_i) \subset \mathcal{B}(p,R).$$

La première inclusion est conséquence du calcul de p_i et R_i. La seconde se traduit par

$$\|f-p_i\|_\infty \leqslant R_i \quad \Rightarrow \quad \|f-p\|_\infty \leqslant \|f-p_i\|_\infty + \|p-p_i\|_\infty \leqslant R_i + \|p-p_i\|_\infty \leqslant R.$$

11. Il devrait être possible de lever cette restriction, voir notamment Kaucher et Miranker [139].

Il suffit pour conclure de calculer $\beta \geqslant \|p - p_i\|_\infty$ et de vérifier que $R_i + \beta \leqslant R$. Comme p est présumé proche du point fixe de τ, on s'attend à ce que le processus réussisse pour i assez grand, pourvu simplement que tous les calculs soient menés à une précision suffisante.

Dans le cas qui nous intéresse, l'application τ étant affine, on a indépendamment de R l'inclusion

$$\tau^i(\mathcal{B}(p,R)) \subset \mathcal{B}(p_i, \gamma_i R + \alpha_i)$$

où γ_i est la constante de Lipschitz apparue dans (9.37) et α_i un terme qui représente les erreurs accumulées au fil du calcul, et peut être rendu arbitrairement petit quitte à augmenter la précision. On renvoie directement la meilleure borne validable par ce procédé (pour i donné), c'est-à-dire

$$R = \frac{\alpha_i + \|p - \hat{p}\|_\infty}{1 - \gamma_i}.$$

9.5.3 Algorithme

La méthode de validation pour une équation (E) d'ordre quelconque est essentiellement celle de la section précédente, appliquée au système du premier ordre compagnon de (E). On considère cette fois l'opérateur

$$\tau: \quad f \mapsto c_{r-1} + \int_0^{\cdot} \left(a_{r-1}\, f^{(r-1)} + \cdots + a_1\, f' + a_0\, f \right)$$

où l'on a posé, pour $k < r-1$,

$$f^{(k)}(x) = c_k + \int_0^x f^{(k+1)} = c_k + \int_0^x \left(\cdots + \int_0^{\cdot} \left(c_{r-2} + \int_0^{\cdot} f^{(r-1)} \right) \cdots \right).$$

Une fonction $y^{(r-1)}$ est point fixe de τ si et seulement si sa primitive $(r-1)$-ième y telle que $y(0) = c_0, \ldots, y^{(r-2)}(0) = c_{r-2}$ satisfait l'équation

$$L \cdot y = y^{(r-1)} - a_{r-1}\, y^{(r-1)} + \cdots + a_1\, y' + a_0\, y = 0$$

et la condition intiale supplémentaire $y^{(r-1)}(0) = c_{r-1}$.

LEMME 9.30. On peut calculer la dérivée ou une primitive d'un polynôme[12] de degré au plus d écrit sur la base de Tchebycheff en $O(d)$ opérations arithmétiques. ♢

DÉMONSTRATION. Si $f = \sum_n c_n T_n$ et $f' = \sum_n c'_n T_n$, on a $2\, n\, c_n = c'_{n-1} + c'_{n+1}$ d'après la relation (9.10) entre les T_n. Les primitives se calculent par application directe de cette formule. Pour dériver f, supposé de degré d, on part du fait que les coefficients c'_n du polynôme dérivé sont nuls lorsque $|n| \geqslant d$, et l'on calcule les coefficients restants, pour $|n|$ décroissant, en utilisant la formule d'intégration vue comme une relation de récurrence inhomogène. □

LEMME 9.31. Soit $f = \sum_{n=-d}^{d} c_n T_n$ un polynôme donné sur la base de Tchebycheff. On peut calculer $M \geqslant 0$ tel que $\|f\|_\infty \leqslant M \leqslant \sqrt{d+1}\, \|f\|_\infty$ en $O(d)$ opérations. ♢

DÉMONSTRATION. On a

$$\|f\|_2 = \left(\frac{1}{\pi} \int_{-1}^{1} \frac{f(t)^2}{\sqrt{1-t^2}}\, \mathrm{d}t \right)^{1/2} = \left(|c_0|^2 + 4 \sum_{n=1}^{d} |c_n|^2 \right)^{1/2}.$$

On dispose ainsi de l'encadrement

$$\|f\|_2 \leqslant \|f\|_\infty \leqslant \sum_{n=-d}^{d} |c_n| \leqslant \sqrt{d+1}\, \|f\|_2$$

12. Attention, l'opération de dérivation ne commute pas avec la troncature des *séries* de Tchebycheff.

où la première inégalité provient de l'écriture intégrale de $\|f\|_2$, la deuxième du fait que $\|T_n\|_\infty \leqslant 1$ pour tout n, et la troisième de l'inégalité de Cauchy-Schwarz. La somme des modules des c_n fournit donc la borne cherchée. □

Algorithme 9.32. ChebErrorBound$\big(L, \big[y(0), y'(0), ..., y^{(r-1)}(0)\big], p\big)$

Entrée. Un opérateur unitaire $L = \partial^r - a_{r-1}(x)\,\partial^{r-1} - \cdots - a_0(x) \in \mathbb{Q}(x)\langle\partial\rangle$ sans pôle pour $x \in [-1;1]$ et des conditions initiales $y^{(k)}(0)$ pour $0 \leqslant k \leqslant r-1$. Un polynôme p de degré d, écrit sur la base de Tchebycheff. Un paramètre de précision $\varepsilon > 0$.

Sortie. Un réel $R > 0$ tel que $\|y - p\|_\infty \leqslant R$, où y désigne la solution de $L \cdot y = 0$ satisfaisant les conditions initiales données.

1 $p_0^{(0)} := p$; $p_0^{(1)} = p'$; ... ; $p_0^{(r-1)} := p^{(r-1)}$

2 calculer $A \geqslant \max_{k=0}^{r-1} \|a_k\|_\infty$ (par exemple, en développant a_k en série de Tchebycheff par l'algorithme 9.28 puis en faisant la somme des coefficients de la série)

3 pour $i = 0, 1, ...$ tant que $\gamma_i := \dfrac{A^i\, e}{i!} \geqslant \dfrac{1}{2}$

  4 pour k de 0 à $r-1$

    5 $q_i^{[k]} := \mathrm{ChebExpandRatpoly}(a_k, p_i^{(k)}, \varepsilon)$ (algorithme 9.28)

  6 $p_{i+1}^{(r-1)} := y^{(r-1)}(0) + \displaystyle\int_0^{\cdot} \Big(q_i^{[r-1]} + q_i^{[r-2]} + \cdots + q_i^{[0]} \Big)$

  7 pour k décroissant de $r-2$ à 0

    8 $p_{i+1}^{(k)} := y^{(k)}(0) + \displaystyle\int_0^{\cdot} p_{i+1}^{(k+1)}$

9 $\alpha := r\, e^{A+1}\, \varepsilon$

10 calculer $\beta_i \geqslant \|p^{(r-1)} - p_i^{(r-1)}\|_\infty$,
par exemple en prenant $\beta_i = \sum_n |c_n|$ où $p^{(r-1)} - p_i^{(r-1)} = \sum_n c_n T_n$

11 renvoyer $\dfrac{\alpha + \beta_i}{1 - \gamma_i}$ ♢

Proposition 9.33. L'algorithme 9.32 s'exécute en $O(d + \log(\varepsilon^{-1}))$ opérations arithmétiques quand $d \to \infty$ et $\varepsilon \to 0$, l'opérateur L étant fixé. ♢

Démonstration. Le nombre d'itérations des boucles ne dépend pas de d. Chaque ligne demande $O(d + \log(\varepsilon^{-1}))$ opérations, d'après la proposition 9.29 et les lemmes 9.30 et 9.31. □

Tant la correction de l'algorithme que l'analyse de la qualité du résultat reposent sur le lemme suivant.

Lemme 9.34. Soit f une fonction analytique sur $[-1;1]$. Les polynômes $p = p_0$ et p_i définis dans l'algorithme 9.32 satisfont

$$\|\tau^i(f^{(r-1)}) - p_i^{(r-1)}\|_\infty \leqslant \gamma_i \left\| f^{(r-1)} - p^{(r-1)} \right\|_\infty + \alpha$$

où

$$\gamma_i = \frac{A^i\, e}{i!}, \qquad \alpha = r\, e^{A+1}\, \varepsilon,$$

et A désigne le maximum des normes $\|a_k\|_\infty$ des coefficients de l'équation. ♢

Démonstration. On pose $f_0 = f$, et pour tout $i \geqslant 0$, on définit f_{i+1} par

$$\begin{cases} f_{i+1}^{(r-1)} = y^{(r-1)}(0) + \displaystyle\int_0^{\cdot} \Big(a_{r-1}\, f_i^{(r-1)} + \cdots + a_0\, f_i \Big) \\ f_{i+1}(x) = y(0) + y'(0)\, x + \cdots + \dfrac{1}{(r-1)!}\, y^{(r-1)}(0)\, x^{r-1} + O(x^r). \end{cases} \tag{9.38}$$

On a donc $f_{i+1}^{(r-1)} = \tau f_i^{(r-1)}$. Montrons par récurrence sur i un résultat un petit peu plus fin que celui de l'énoncé, à savoir que pour tout $x \in [-1;1]$,

$$|f_i^{(r-1)}(x) - p_i^{(r-1)}(x)| \leqslant \gamma_i(x)\, \|f^{(r-1)} - p^{(r-1)}\|_\infty + \alpha_i(x) \tag{9.39}$$

où l'on a posé

$$\begin{aligned} R &= \|f^{(r-1)} - p^{(r-1)}\|_\infty \\ \gamma_i(x) &= \sum_{0 \leqslant k_1,\dots,k_i < r} \frac{\|a_{k_1}\|_\infty \cdots \|a_{k_i}\|_\infty\, |x|^{ri-k_1-\cdots-k_i}}{(ri-k_1-\cdots-k_i)!}, \\ &\qquad\qquad \text{(avec } \gamma_0(x) = \textstyle\sum_{\{*\}} 1 = 1\text{)} \\ \alpha_i(x) &= r\,\varepsilon \int_0^x (\gamma_0 + \cdots + \gamma_{i-1}). \end{aligned}$$

L'inégalité (9.39) est immédiate pour $i=0$. Supposons-la établie pour un certain i. En comparant la définition de f_{i+1} par (9.38) à celle de p_{i+1} à l'étape 6 de l'algorithme, on écrit

$$f_{i+1}^{(r-1)} - p_{i+1}^{(r-1)} = \int_0^{\cdot} \Big(\Big(a_{r-1}\, f_i^{(r-1)} - q_i^{[r-1]} \Big) + \cdots + \Big(a_0\, f_i^{(0)} - q_i^{[0]} \Big) \Big),$$

d'où

$$|f_{i+1}^{(r-1)} - p_{i+1}^{(r-1)}| \leqslant \int_0^{\cdot} \sum_{k=0}^{r-1} \Big(|a_k|\, |f_i^{(k)} - p_i^{(k)}| + |a_k\, p_i^{(k)} - q_i^{[k]}| \Big).$$

Puisque les valeurs en zéro des r premières dérivées de f_i et p_i coïncident, l'hypothèse de récurrence fournit l'inégalité

$$\begin{aligned} \int_0^{\cdot} |f_i^{(k)} - p_i^{(k)}| &\leqslant \left(\int_0^{\cdot} \right)^{r-k} \big(\gamma_i\, R + \alpha_i \big) \\ &= \left(\int_0^{\cdot} \right)^{r-k} \gamma_i\, R + \left(\int_0^{\cdot} \right)^{r-k+1} (\gamma_0 + \cdots + \gamma_{i-1}). \end{aligned}$$

Par ailleurs on a $\|a_k\, p_i^{(k)} - q_i^{[k]}\|_\infty \leqslant \varepsilon$ d'après la spécification de l'algorithme 9.28. En observant que

$$\sum_{k=0}^{r-1} \|a_k\|_\infty \left(\int_0^{\cdot} \right)^{r-k} \gamma_i = \gamma_{i+1}.$$

et en sommant sur k, il vient

$$\begin{aligned} |f_{i+1}^{(r-1)}(x) - p_{i+1}^{(r-1)}(x)| &\leqslant \sum_{k=0}^{r-1} \left(\|a_k\|_\infty \int_0^x |f_i^{(k)} - p_i^{(k)}| + \int_0^x |a_k\, p_i^{(k)} - q_i^{[k]}| \right) \\ &\leqslant \gamma_{i+1}(x)\, R + r\,\varepsilon \int_0^x (\gamma_1 + \cdots + \gamma_i) + r\,\varepsilon\, |x| \\ &= \gamma_{i+1}(x)\, R + \alpha_{i+1}(x), \end{aligned}$$

qui est la borne cherchée.

Pour conclure, il reste à borner $\gamma_i(x)$ par une expression plus simple. On écrit

$$\begin{aligned} \sum_{k_1,\dots,k_i \geqslant 0} \frac{t^{k_1+\cdots+k_i}}{(i+k_1+\cdots+k_i)!} &= \sum_{n=0}^{\infty} \frac{\#\{(k_1,\dots,k_i) : k_1+\cdots+k_i = n\}}{(i+n)!}\, t^n \\ &= \sum_{n=0}^{\infty} \binom{n+i-1}{i-1} \frac{t^n}{(n+i)!} = \sum_{n=0}^{\infty} \frac{t^n}{(n+i)\, n!\, (i-1)!} \end{aligned}$$

d'où

$$\gamma_i(x) \leqslant \sum_{0 \leqslant k_1,\dots,k_i < r} \frac{A^i\, |x|^{ri-k_1-\cdots-k_i}}{(ri-k_1-\cdots-k_i)!} = \sum_{0 \leqslant k_1,\dots,k_i < r} \frac{A^i\, |x|^{i+k_1+\cdots+k_i}}{(i-k_1+\cdots+k_i)!} \leqslant \frac{(A\,|x|)^i\, e^{|x|}}{i!}$$

et $\alpha_i(x) \leqslant r\,\varepsilon\, e^{|x|} \exp_{;i}(A\,|x|) \leqslant r\,\varepsilon\, e^{(A+1)|x|}$. □

Proposition 9.35. Avec les notations de l'algorithme 9.32, soit y l'unique solution de l'équation différentielle $L\cdot y = 0$ satisfaisant les conditions initiales données en entrée. La valeur R renvoyée par l'algorithme est une borne supérieure sur $\|p-y\|_\infty$. ♢

Démonstration. D'après le lemme 9.34, si $\|f^{(r-1)} - p^{(r-1)}\|_\infty \leqslant R$, alors on a

$$\begin{aligned}\|\tau^i(f^{(r-1)}) - p^{(r-1)}\|_\infty &\leqslant \|\tau^i(f) - p_i^{(r-1)}\|_\infty + \|p_i^{(r-1)} - p^{(r-1)}\|_\infty \\ &\leqslant \gamma_i\, R + \alpha + \beta_i = \gamma_i \frac{\alpha+\beta_i}{1-\gamma_i} + \alpha + \beta_i = R.\end{aligned}$$

Autrement dit, avec $\mathcal{B}(p^{(r-1)}, R)$ définie par l'équation (9.36), on a l'inclusion

$$\tau\big(\mathcal{B}(p^{(r-1)}, R)\big) \subset \mathcal{B}(p^{(r-1)}, R).$$

La boule $\mathcal{B}(p^{(r-1)}, R)$ est un espace métrique complet, et la preuve du lemme 9.34 où l'on prend $\varepsilon = 0$ et remplace p par une fonction analytique quelconque g montre que l'opérateur τ^i est γ_i-lipschitzien pour tout i, donc contractant à l'issue de la boucle[13]. D'après le théorème du point fixe de Banach, la restriction de τ à $\mathcal{B}(p^{(r-1)}, R)$ admet un point fixe, qui ne peut être que $y^{(r-1)}$. Ainsi, le polynôme p satisfait

$$\|y^{(r-1)} - p^{(r-1)}\|_\infty \leqslant R,$$

d'où le résultat en intégrant $r-1$ fois. □

9.5.4 Finesse du résultat

Il reste à évaluer à quel point la borne renvoyée par l'algorithme 9.32 est pessimiste par rapport à la véritable valeur de $\|y-p\|_\infty$. Pour d grand, et en supposant pour fixer les idées que $\|y-p\|_\infty = 2^{-\Theta(d)}$, elle est en fait très proche de l'optimum.

Proposition 9.36. Le résultat de l'algorithme 9.32 vérifie

$$R \leqslant \sqrt{d+1}\,\big(3\,\|y^{(r-1)} - p^{(r-1)}\|_\infty + O(\varepsilon)\big)$$

quand $\varepsilon \to 0$, où la constante cachée par le $O(\cdot)$ ne dépend que de L. ♢

Démonstration. Appliquons le lemme 9.34 à $f = y$, la solution de l'équation : il vient

$$\|y^{(r-1)} - p_i^{(r-1)}\|_\infty = \|\tau^i\cdot y^{(r-1)} - p_i^{(r-1)}\|_\infty \leqslant \gamma_i\,\|y^{(r-1)} - p^{(r-1)}\|_\infty + \alpha$$

et donc

$$\|p_0^{(r-1)} - p_i^{(r-1)}\|_\infty \leqslant (1+\gamma_i)\,\|y^{(r-1)} - p^{(r-1)}\|_\infty + \alpha.$$

D'après le lemme 9.31, on peut prendre

$$\beta_i \leqslant \sqrt{d+1}\,\|p_0^{(r-1)} - p_i^{(r-1)}\|_\infty.$$

La borne renvoyée satisfait donc

$$R = \frac{\alpha+\beta_i}{1-\gamma_i} \leqslant 2\,(1+\sqrt{d+1})\,\alpha + 3\,\sqrt{d+1}\,\|y^{(r-1)} - p^{(r-1)}\|_\infty.$$

On a $\alpha = O(\varepsilon)$ d'après l'étape 9 de l'algorithme. □

Le terme α de la borne peut être pris arbitrairement petit, et $\gamma_i \to 0$ quand $i \to \infty$. La partie la plus pessimiste du calcul est la méthode naïve de calcul de β issue du

13. On pourrait contourner cet argument en invoquant le théorème du point fixe de Schauder.

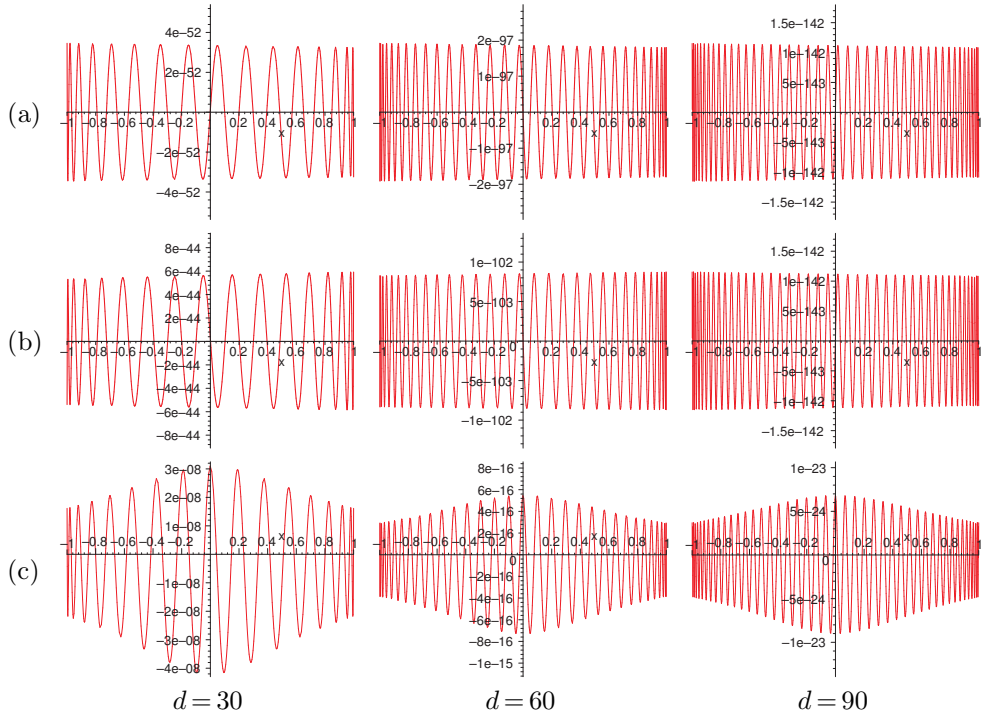


Figure 9.3. Graphes des différences $y - p$ entre approximations p de degré $d \in \{30, 60, 90\}$ calculées par l'algorithme 9.16 pour les problèmes de l'exemple 9.37 et solutions exactes y correspondantes. Les oscillations d'amplitude presque uniforme des deux premières lignes sont caractéristiques des approximations polynomiales proches de l'optimum, en vertu du théorème d'oscillation de la Vallée Poussin [47, §3.4].

lemme 9.31. Quitte à la remplacer par un algorithme (potentiellement plus coûteux) capable de majorer $\|p_0^{(r-1)} - p_i^{(r-1)}\|_\infty$ avec une erreur arbitrairement petite, on peut prouver des bornes d'erreur arbitrairement proches de $\|y^{(r-1)} - p^{(r-1)}\|_\infty$.

9.6 Expériences

Nous avons implémenté en Maple une variante des algorithmes de ce chapitre qui mélange arithmétique rationnelle et arithmétique d'intervalles. Les calculs par intervalles reposent sur le module intpakX [119]. Notre implémentation, à l'état de prototype, n'est pas encore disponible publiquement. La procédure de validation implémentée pour l'instant diffère légèrement de celle décrite ci-dessus (et produit vraisemblablement des bornes un peu plus grossières). Pour toutes ces raisons, nous nous limitons ici à illustrer sur quelques exemples simples la qualité des approximations et des bornes d'erreur que fournissent les méthodes de ce chapitre.

EXEMPLE 9.37. Pour chacun des exemples suivants, la figure 9.3 présente l'allure de la différence entre la valeur approchée p de la solution donnée par l'algorithme de la section 9.3.1 et la solution exacte. Le tableau 9.1 compare la borne d'erreur validée à partir de l'équation différentielle, l'erreur réellement atteinte par p et l'erreur optimale possible avec un polynôme du même degré. Pour les besoins de l'expérience, le paramètre ε de l'algorithme 9.32 est choisi de sorte que α soit de l'ordre de l'erreur exacte $\|p - y\|_\infty$.

(a) Le premier exemple est adapté de Kaucher et Miranker [139, p. 222]. Il s'agit de la fonction

$$y(x) = \frac{e^{x/2}}{\sqrt{x+16}}$$

solution de l'équation différentielle du premier ordre

$$2\,(x+16)\,y'(x) = (x+15)\,y(x), \qquad y(0) = \frac{1}{4}.$$

(b) Le problème suivant, emprunté à Geddes [105, p. 31],

$$y^{(4)}(x) = y(x), \qquad y(0) = \frac{3}{2}, \quad y'(0) = -\frac{1}{2}, \quad y''(0) = -\frac{3}{2}, \quad y'''(0) = \frac{1}{2}$$

est une équation d'ordre quatre sans singularité finie, de solution exacte

$$y(x) = \frac{3}{2}\cos x - \frac{1}{2}\sin x.$$

(c) Enfin, l'équation d'ordre un

$$(1+2\,x^2)^2\,y'(x) + 4\,x\,y(x) = 0, \qquad y(0) = e,$$

possède des points singuliers imaginaires $x = \pm i\,\sqrt{2}/2$ relativement proches du segment $[-1;1]$ et la solution exacte

$$y(x) = \exp\frac{1}{1+2\,x^2}.$$

Notons aussi que sa condition initiale n'est pas représentable exactement en arithmétique rationnelle, ce qui rend indispensable l'utilisation d'un minimum d'arithmétique d'intervalles.

On constate que $\|p-y\|_\infty$ est extrêmement proche de l'erreur optimale accessible avec un polynôme de même degré, voire en est indistingable à la précision du tableau. La borne prouvée R est un peu plus pessimiste, surtout quand l'ordre augmente, mais l'écart ne dépasse pas quelques chiffres significatifs. La proposition 9.36 explique un facteur pouvant aller jusqu'à 15 (pour le degré 30) ou 30 (pour le degré 90) environ entre R et $\|p-y\|_\infty$. On observe pourtant une surestimation plus importante, qui s'explique par les petites différences entre la méthode décrite ci-dessus et celle utilisée dans l'implémentation, et notamment par les erreurs accumulées lors des calculs par intervalles. La borne anormalement élevée de la toute première ligne mériterait plus ample investigation. ♢

9.7 Perspectives

Les propositions 9.19 et 9.36, couplées aux expériences rapportées dans la section précédente, suggèrent que les outils de chapitre sont capables de produire des approximations de très bonne qualité des fonctions D-finies analytiques sur un segment. Ce travail a tout de même un certain goût d'inachevé.

En premier lieu, il manque manifestement un théorème qui synthétise ce que l'on obtient en combinant convenablement les deux principaux algorithmes de ce chapitre. Dans l'idéal, on pourrait espérer un résultat du type « étant donnés y, d et ε, on peut calculer un polynôme p et une borne R avec $\|y-p\|_\infty \leqslant R \leqslant K_d\,\|y-p^*\|_\infty + \varepsilon$, en complexité arithmétique linéaire en d », où K_d ne dépend que de d (mais la constante cachée par le $O(\cdot)$ dépend de y) ; voire un énoncé similaire sans le terme en ε. Plusieurs obstacles techniques subsistent cependant. Toujours sur le plan théorique, il serait souhaitable de se débarrasser ensuite des hypothèses **H1** et **H2** introduites pour l'analyse du calcul des coefficients.

Du point de vue pratique, notre code doit devenir plus robuste, plus rapide, et coller de plus près à l'algorithme théorique. Cela permettra de pousser plus loin les expériences (assez minimales !) présentées ici, et de l'appliquer à des besoins comme le tracé des graphes apparaissant dans le DDMF déjà mentionné au chapitre 2. Nous espérons en diffuser prochainement une version utilisable par autrui.

	degré d	borne R	$\|y-p\|_\infty$	$\|y-p^*\|$	$\log_{10} R/\|y-p\|_\infty$
(a) $\frac{e^{x/2}}{\sqrt{x+16}}$	30	$2{,}0\cdot10^{-47}$	$3{,}4\cdot10^{-52}$	$3{,}4\cdot10^{-52}$	4,8
	60	$7{,}3\cdot10^{-97}$	$1{,}9\cdot10^{-97}$	$1{,}9\cdot10^{-97}$	0,58
	90	$4{,}4\cdot10^{-141}$	$1{,}2\cdot10^{-141}$	$1{,}1\cdot10^{-142}$	0,57
(b) $\frac{3\cos x-\sin x}{2}$	30	$7{,}8\cdot10^{-41}$	$5{,}9\cdot10^{-44}$	$5{,}6\cdot10^{-44}$	3,1
	60	$4{,}5\cdot10^{-99}$	$8{,}7\cdot10^{-103}$	$8{,}5\cdot10^{-103}$	3,7
	90	$3{,}5\cdot10^{-164}$	$3{,}0\cdot10^{-168}$	$3{,}0\cdot10^{-168}$	4,1
(c) $e^{\,1/(1+2x^2)}$	30	$1{,}5\cdot10^{-7}$	$4{,}1\cdot10^{-8}$	$2{,}8\cdot10^{-8}$	0,57
	60	$2{,}6\cdot10^{-15}$	$7{,}3\cdot10^{-16}$	$4{,}9\cdot10^{-16}$	0,56
	90	$3{,}2\cdot10^{-23}$	$9{,}0\cdot10^{-24}$	$6{,}0\cdot10^{-24}$	0,56

Tableau 9.1. Qualité des bornes validées par une variante de l'algorithme 9.32 pour les problèmes de l'exemple 9.37. La colonne $\|y-p\|_\infty$ indique la véritable erreur maximale entre l'approximant p et la fonction y (celle-là même qu'on peut lire sur les graphes de la figure 9.3). La colonne $\|y-p^*\|_\infty$ donne une valeur approchée de l'erreur commise par le meilleur approximant de degré d de y, calculée grâce à Sollya [52].

Enfin, à un peu plus long terme, il serait naturel d'examiner les généralisations possibles aux développements sur d'autres familles de polynômes orthogonaux, notamment le reste de la classe des polynômes de Gegenbauer. Autre question intéressante : à quelle complexité peut-on aboutir pour traiter l'analogue de la question 9.1 dans le cas des équations différentielles non linéaires ? On peut s'attendre à une algorithmique différente, fondée sur la méthode de Newton et non sur une relation de récurrence, mais certaines idées de ce chapitre demeurent peut-être applicables.

Bibliographie

[1] *Proceedings of the 1977 MACSYMA User's Conference*, juillet 1977.
http://dspace.udel.edu:8080/dspace/handle/19716/1398

[2] *Séries divergentes et procédés de resommation – Journées X-UPS 1991*, 1991.

[3] Digital library of mathematical functions, 2010. Companion to the NIST Handbook of Mathematical Functions [195].
http://dlmf.nist.gov/

[4] J. Abad, F. J. Gómez et J. Sesma. An algorithm to obtain global solutions of the double confluent Heun equation. *Numerical Algorithms*, 49(1-4):33–51, 2008.
http://arxiv.org/abs/0811.1172

[5] J. Abdeljaoued et H. Lombardi. *Méthodes matricielles — Introduction à la complexité algébrique*. Springer, 2004.

[6] S. A. Abramov, M. Bronstein et D. E. Khmelnov. On regular and logarithmic solutions of ordinary linear differential systems. In V. G. Ganzha, E. W. Mayr et E. V. Vorozhtsov, éditeurs, *CASC*, volume 3718 de *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005.
http://www.ccas.ru/zavar/abrsa/ps/abkCASC5.ps

[7] M. Abramowitz et I. A. Stegun. *Handbook of Mathematical Functions*. Number 55 in National Bureau of Standards Applied Mathematics Series. National Bureau of Standards, 1964. Dover reprint, 1972.
http://people.math.sfu.ca/~cbm/aands/

[8] R. Apéry. Irrationalité de $\zeta(2)$ et $\zeta(3)$. *Astérisque*, 61:11–13, 1979.

[9] J. Arndt. *Matters Computational. Ideas, Algorithms, Source Code*. Springer, 2011.
http://www.jjj.de/fxt/fxtbook.pdf.gz

[10] D. Bailey, R. Barrio et J. Borwein. High-precision computation: Mathematical physics and dynamics, 2011.
http://crd.lbl.gov/~dhbailey/dhbpapers/hpmpd.pdf

[11] C. Banderier, F. Chern et H.-K. Hwang. Asymptotics of D-finite sequences. En préparation.

[12] M. Barkatou, F. Chyzak et M. Loday-Richaud. Remarques algorithmiques liées au rang d'un opérateur différentiel linéaire. In F. Fauvet et C. Mitschi, éditeurs, *From Combinatorics to Dynamical Systems*, volume 3 de *IRMA Lectures in Mathematics and Theoretical Physics*, pages 87–129, 2003.
http://algo.inria.fr/papers/pdf/BaChLR03.pdf

[13] M. Barkatou et E. Pflügel. An algorithm computing the regular formal solutions of a system of linear differential equations. *Journal of Symbolic Computation*, 28(4-5):569–587, 1999.

[14] D. Barton, I. Willers et R. Zahar. The automatic solution of systems of ordinary differential equations by the method of Taylor series. *The Computer Journal*, 14(3):243, 1971.

[15] M. Beeler, R. W. Gosper et R. Schroeppel. Hakmem. AI Memo 239, MIT Artificial Intelligence Laboratory, 1972.
http://hdl.handle.net/1721.1/6086

[16] P. Bell, J. Delvenne, R. Jungers et V. Blondel. The continuous Skolem-Pisot problem. *Theoretical Computer Science*, 411(40-42):3625–3634, 2010.
http://www.inma.ucl.ac.be/~blondel/publications/10BDJ.pdf

[17] F. Bellard. Computation of 2700 billion decimal digits of Pi using a desktop computer, janvier 2010.
http://bellard.org/pi/pi2700e9/

[18] A. Benoit. Thèse de doctorat, École polytechnique. En préparation.

[19] A. Benoit, F. Chyzak, A. Darrasse, S. Gerhold, M. Mezzarobba et B. Salvy. The dynamic dictionary of mathematical functions (DDMF). In *Mathematical Software - ICMS 2010* [103], pages 35–41.
http://marc.mezzarobba.net/ecrits/BenoitEtAl_DDMF_2010.pdf

[20] A. Benoit, M. Joldeş et M. Mezzarobba. Rigorous uniform approximation of D-finite functions using Chebyshev expansions. En préparation.

[21] A. Benoit et B. Salvy. Chebyshev expansions for solutions of linear differential equations. In *ISSAC '09* [135], pages 23–30.
http://hal.archives-ouvertes.fr/docs/00/39/57/16/PDF/arxiv.pdf

[22] S. Berjeaut. Le Minotaure, 2009.
http://www.myspace.com/video/simon-berjeaut/le-minotaure-par-simon-berjeaut/5281150 3

[23] D. J. Bernstein. Multidigit multiplication for mathematicians. 2001.
http://cr.yp.to/papers/m3.pdf

[24] D. J. Bernstein. Fast multiplication and its applications. In J. Buhler et P. Stevenhagen, éditeurs, *Algorithmic Number Theory*, pages 325–384. Cambridge University Press, 2008.
http://www.msri.org/communications/books/Book44/

[25] F. Beukers. G-function. In *Encyclopaedia of Mathematics* [125].
http://eom.springer.de/

[26] W. Bickley, L. Comrie, J. Miller, D. Sadler et A. Thompson. *Bessel functions. Part II. Functions of positive integer order*, volume X de *Mathematical Tables*. British Association for the Advancement of Science, 1952.

[27] D. Bini et G. Fiorentino. Design, analysis, and implementation of a multiprecision polynomial rootfinder. *Numerical Algorithms*, 23(2):127–173, 2000.

[28] G. D. Birkhoff et W. J. Trjitzinsky. Analytic theory of singular difference equations. *Acta Mathematica*, 60:1–89, 1932.

[29] V. D. Blondel et N. Portier. The presence of a zero in an integer linear recurrent sequence is NP-hard to decide. *Linear Algebra and its Applications*, 351/352:91–98, 2002.
http://www.inma.ucl.ac.be/publi/223540.pdf

[30] J. Borwein et R. Crandall. Closed forms: what they are and why we care. *Notices of the American Mathematical Society*, 2012. À paraître.
http://carma.newcastle.edu.au/~jb616/closed-form.pdf

[31] J. M. Borwein et P. B. Borwein. *Pi and the AGM*. Wiley, 1987.

[32] P. B. Borwein. On the complexity of calculating factorials. *Journal of Algorithms*, 6:376–380, 1985.

[33] A. Bostan, F. Chyzak, M. Giusti, R. Lebreton, B. Salvy et É. Schost. Algorithmes efficaces en calcul formel. Notes du cours 2-22 du Master parisien de recherche en informatique, 2010–2011. Version du 21 février 2011.
http://www.mpri.master.univ-paris7.fr/C-2-22.html

[34] A. Bostan, T. Cluzeau et B. Salvy. Fast algorithms for polynomial solutions of linear differential equations. In M. Kauers, éditeur, *ISSAC '05*, pages 45–52. ACM, 2005.
http://algo.inria.fr/papers/pdf/BoClSa05.pdf

[35] A. Bostan, B. Salvy et Éric Schost. Power series composition and change of basis. In J. R. Sendra et L. González-Vega, éditeurs, *ISSAC '08*, pages 269–276. ACM, 2008.
http://hal.archives-ouvertes.fr/inria-00273385/

[36] J. Boyd. *Chebyshev and Fourier spectral methods*. Dover, 2000.
http://www-personal.umich.edu/~jpboyd/BOOK_Spectral2000.html

[37] R. P. Brent. The complexity of multiple-precision arithmetic. In R. S. Anderssen et R. P. Brent, éditeurs, *The Complexity of Computational Problem Solving*, pages 126–165, Brisbane, Australia, 1976.
http://wwwmaths.anu.edu.au/~brent/pub/pub032.html

[38] R. P. Brent. Fast multiple precision evaluation of elementary functions. *Journal of the ACM*, 23:242–251, 1976.
http://wwwmaths.anu.edu.au/~brent/pub/pub034.html

[39] R. P. Brent. A Fortran multiple-precision arithmetic package. *ACM Transactions on Mathematical Software*, 4(1):57–70, mars 1978.
http://maths.anu.edu.au/~brent/pub/pub042.html

[40] R. P. Brent et P. Zimmermann. *Modern Computer Arithmetic*. Cambridge University Press, 2010.
http://www.loria.fr/~zimmerma/mca/mca-cup-0.5.7.pdf

[41] N. Brisebarre et M. Joldeş. Chebyshev interpolation polynomial-based tools for rigorous computing. In *ISSAC '10* [148].
http://prunel.ccsd.cnrs.fr/ensl-00472509/en/

[42] M. Bronstein. *Symbolic integration I*, volume 1 de *Algorithms and Computation in Mathematics*. Springer-Verlag, second edition, 2005.

[43] M. Bronstein et B. Salvy. Full partial fraction decomposition of rational functions. In M. Bronstein, éditeur, *ISSAC '93*, pages 157–160. ACM, 1993.
http://algo.inria.fr/papers/other/fullparfrac.ps.Z

[44] P. Bürgisser, M. Clausen et M. A. Shokrollahi. *Algebraic complexity theory*, volume 315 de *Grundlagen der mathematischen Wissenschaften*. Springer, 1997.

[45] D. G. Cantor et E. Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28(7):693–701, 1991.
http://www4.ncsu.edu/~kaltofen/bibliography/91/CaKa91.pdf

[46] C. Chabaud. *Séries génératrices algébriques : asymptotique et applications combinatoires*. Thèse de doctorat, Université Pierre et Marie Curie, novembre 2002.

[47] E. Cheney. *Introduction to approximation theory*. American Mathematical Society, 1998.

[48] H. Cheng, B. Gergel, E. Kim et E. Zima. Space-efficient evaluation of hypergeometric series. *SIGSAM Bulletin*, 39(2):41–52, juin 2005.
`http://sigsam.org/bulletin/articles/152/Cheng.pdf`

[49] H. Cheng, G. Hanrot, E. Thomé, E. Zima et P. Zimmermann. Time- and space-efficient evaluation of some hypergeometric constants. In D. Wang, éditeur, *ISSAC '07*. ACM, 2007.
`http://www.cs.uleth.ca/~cheng/papers/issac2007.pdf`

[50] S. Chevillard. *Évaluation efficace de fonctions numériques. Outils et exemples*. Thèse de doctorat, École normale supérieure de Lyon, 2009.
`http://tel.archives-ouvertes.fr/tel-00460776`

[51] S. Chevillard, J. Harrison, M. Joldeş et C. Lauter. Efficient and accurate computation of upper bounds of approximation errors. *Theoretical Computer Science*, 16(412):1523–1543, 2011.
`http://hal.archives-ouvertes.fr/ensl-00445343/`

[52] S. Chevillard, M. Joldeş et C. Lauter. Sollya: An environment for the development of numerical codes. In *Mathematical Software - ICMS 2010* [103], pages 28–31.
`http://www-sop.inria.fr/members/Sylvain.Chevillard/download/papers/ChevillardJoldesLauterICMS2010.pdf`

[53] D. V. Chudnovsky et G. V. Chudnovsky. On expansion of algebraic functions in power and Puiseux series, I. *Journal of Complexity*, 2(4):271–294, 1986.

[54] D. V. Chudnovsky et G. V. Chudnovsky. Computer assisted number theory with applications. In *Number theory (New York, 1984–1985)*, volume 1240 de *Lecture Notes in Mathematics*, pages 1–68. Springer, 1987.

[55] D. V. Chudnovsky et G. V. Chudnovsky. On expansion of algebraic functions in power and Puiseux series, II. *Journal of Complexity*, 3(1):1–25, 1987.

[56] D. V. Chudnovsky et G. V. Chudnovsky. Approximations and complex multiplication according to Ramanujan. In *Ramanujan revisited*, pages 375–472. Academic Press, 1988.

[57] D. V. Chudnovsky et G. V. Chudnovsky. Computer algebra in the service of mathematical physics and number theory. In *Computers in Mathematics* [58], pages 109–232.

[58] D. V. Chudnovsky et R. D. Jenks, éditeurs. *Computers in Mathematics*, volume 125 de *Lecture Notes in Pure and Applied Mathematics*, Proceedings of the International Conference on Computers and Mathematics, Stanford University, 1986. Dekker, 1990.

[59] F. Chyzak. Groebner bases, symbolic summation and symbolic integration. In B. Buchberger et F. Winkler, éditeurs, *Groebner Bases and Applications (Proc. of the Conference 33 Years of Gröbner Bases)*, volume 251 de *London Mathematical Society Lecture Notes Series*, pages 32–60. Cambridge University Press, 1998.
`http://hal.archives-ouvertes.fr/inria-00073391/`

[60] F. Chyzak. An extension of Zeilberger's fast algorithm to general holonomic functions. *Discrete Mathematics*, 217(1-3):115–134, 2000.

[61] F. Chyzak et A. Darrasse. Dynamow, 2008–2011.
`http://ddmf.msr-inria.inria.fr/DynaMoW/`

[62] F. Chyzak et A. Darrasse. Using Camlp4 for presenting dynamic mathematics on the web: DynaMoW, an OCaml language extension for the run-time generation of mathematical contents and their presentation on the web. An experience report. In *ICFP 2011*, 2011. To appear.
`http://ddmf.msr-inria.inria.fr/download/DynaMoW-ICFP11.pdf`

[63] F. Chyzak, P. Dumas, H. Le, J. Martins, M. Mishna et B. Salvy. Taming apparent singularities via Ore closure. In preparation.

[64] F. Chyzak, M. Kauers et B. Salvy. A non-holonomic systems approach to special function identities. In *ISSAC '09* [135].
`http://www.risc.jku.at/people/mkauers/publications/chyzak09a.pdf`

[65] C. Clenshaw. A note on the summation of Chebyshev series. *Mathematics of Computation*, 9:118–120, 1955.
`http://www.ams.org/journals/mcom/1955-09-051/S0025-5718-1955-0071856-0/`

[66] C. W. Clenshaw. The numerical solution of linear differential equations in Chebyshev series. *Proceedings of the Cambridge Philosophical Society*, 53(1):134–149, 1957.

[67] E. Coddington et N. Levinson. *Theory of Ordinary Differential Equations*. McGraw-Hill, 1955.

[68] Computation Laboratory. *Tables of the error function and its derivative.* US National Bureau of Standards, 1954. A reissue of Mathematical table 8, Tables of probability functions, v. 1, published in 1941 by the Laboratory under its earlier name: Mathematical Tables Project.
`http://www.bookprep.com/read/mdp.39015067081037`

[69] D. Coppersmith et S. Winograd. Matrix multiplication via arithmetical progressions. *Journal of Symbolic Computation*, 9:251–280, 1990.

[70] G. Corliss. Survey of interval algorithms for ordinary differential equations. *Applied Mathematics and Computation*, 31:112–120, 1989.

[71] G. Corliss et Y. Chang. Solving ordinary differential equations using Taylor series. *ACM Transactions on Mathematical Software*, 8(2):114–144, 1982.
ftp://ftp.cdf.toronto.edu/pub/consens/publicationsUpTo1995/sigmod94.ps.Z

[72] G. F. Corliss. Guaranteed error bounds for ordinary differential equations. pages 1–75. Oxford University Press, 1995. Lecture notes for a sequence of five lectures at the VI-th SERC Numerical Analysis Summer School, Leicester University, 25 - 29 July, 1994.

[73] R. Crandall et C. Pomerance. *Prime Numbers. A Computational Perspective*. Springer, 2nd edition, 2005.

[74] J. Denef et L. Lipshitz. Decision problems for differential equations. *Journal of Symbolic Logic*, 54(3):941–950, sep 1989.

[75] D. Diderot et J. l. R. d'Alembert, éditeurs. *Encyclopédie, ou Dictionnaire raisonné des sciences, des arts et des métiers*, volume 2. Briasson, David, Le Breton, Durand, 1752.
http://commons.wikimedia.org/wiki/Encyclopédie,_ou_Dictionnaire_raisonné_des_sciences,_des_arts_et_des_métiers

[76] J. Dieudonné. *Calcul infinitésimal*. Hermann, 2e edition, 1980.

[77] C.-É. Drevet, M. N. Islam et É. Schost. Optimization techniques for small matrix multiplication, mai 2010.
http://www.csd.uwo.ca/~eschost/publications/DrIsSc09.pdf

[78] T. Driscoll, F. Bornemann et L. Trefethen. The chebop system for automatic solution of differential equations. *BIT Numerical Mathematics*, 48(4):701–723, 2008.
https://people.maths.ox.ac.uk/trefethen/publication/PDF/2008_131.pdf

[79] Z. Du et C. Yap. Uniform complexity of approximating hypergeometric functions with absolute error. In S.-I. Pae et H. Park, éditeurs, *ASCM 2005*, pages 246–249. Korea Institute for Advanced Study, 2005.
http://www.cs.nyu.edu/exact/doc/approxHyper.pdf

[80] J.-G. Dumas, P. Giorgi et C. Pernet. Dense linear algebra over word-size prime fields: the FFLAS and FFPACK packages. *ACM Transactions on Mathematical Software*, 35(3), 2008.
http://hal.archives-ouvertes.fr/hal-00018223/

[81] R. Dupont. *Moyenne arithmético-géométrique, suites de Borchardt et applications*. Thèse de doctorat, École polytechnique, Palaiseau, 2006.
http://www.lix.polytechnique.fr/Labo/Regis.Dupont/these_soutenance.pdf

[82] I. Eble et M. Neher. ACETAF: A software package for computing validated bounds for Taylor coefficients of analytic functions. *ACM Transactions on Mathematical Software*, 29(3):263–286, 2003.
http://iamlasun8.mathematik.uni-karlsruhe.de/~ae16/acetaf.html

[83] T. H. Einwohner et R. J. Fateman. A MACSYMA package for the generation and manipulation of Chebyshev series. In G. H. Gonnet, éditeur, *ISSAC '89*, pages 180–185. ACM, 1989.
http://http.cs.berkeley.edu/~fateman/papers/cheby-89.pdf

[84] M. K. El-Daou, E. L. Ortiz et H. Samara. A unified approach to the tau method and Chebyshev series expansion techniques. *Computers & Mathematics with Applications*, 25(3):73–82, 1993.

[85] C. Epstein, W. Miranker et T. Rivlin. Ultra-arithmetic I: function data types. *Mathematics and Computers in Simulation*, 24(1):1–18, 1982.

[86] C. Epstein, W. Miranker et T. Rivlin. Ultra-arithmetic II: intervals of polynomials. *Mathematics and Computers in Simulation*, 24(1):19–29, 1982.

[87] A. Erdélyi, W. Magnus, F. Oberhettinger et F. Tricomi. *Higher Transcendental Functions*. Krieger, 1981. 3 volumes.

[88] G. Estrin. Organization of computer systems – the fixed plus variable structure computer. In *Proceedings of the Western Joint IRE-AIEE-ACM Computer Conference*, pages 33–40. ACM, 1960.

[89] P. Falloon, P. Abbott et J. Wang. Theory and computation of spheroidal wavefunctions. *Journal of Physics A: Mathematical and General*, 36:5477–5495, 2003.
http://arxiv.org/abs/math-ph/0212051

[90] J. D. Farmer et S. C. Leth. An asymptotic formula for powers of binomial coefficients. *The Mathematical Gazette*, 89(516):385–391, 2005.

[91] P. Flajolet et A. M. Odlyzko. Singularity analysis of generating functions. *SIAM Journal on Discrete Mathematics*, 3(2):216–240, 1990.
http://algo.inria.fr/flajolet/Publications/FlOd90b.pdf

[92] P. Flajolet et C. Puech. Partial match retrieval of multidimensional data. *Journal of the ACM*, 33(2):371–407, 1986.
http://algo.inria.fr/flajolet/Publications/FlPu86.pdf

[93] P. Flajolet et B. Salvy. Computer algebra libraries for combinatorial structures. *Journal of Symbolic Computation*, 20(5-6):653–671, 1995.
http://hal.inria.fr/inria-00074178_v1/

[94] P. Flajolet et R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.
http://algo.inria.fr/flajolet/Publications/book.pdf

[95] A. R. Forsyth. *Theory of differential equations*, volume 4. Cambridge University Press, 1902. Part III: Ordinary Linear Equations.
http://www.archive.org/details/theorydifferent08forsgoog

[96] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier et P. Zimmermann. MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Transactions on Mathematical Software*, 33(2):13:1–13:15, juin 2007.
http://www.mpfr.org/

[97] L. Fox. Chebyshev methods for ordinary differential equations. *The Computer Journal*, 4(4):318, 1962.

[98] L. Fox et I. Parker. *Chebyshev polynomials in numerical analysis*. Oxford University Press, 1968.

[99] Free Software Foundation. GNU Lesser general public license, version 2.1, février 1999.
http://www.gnu.org/licenses/lgpl-2.1.html

[100] M. Frigo et S. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.
http://fftw.org/fftw-paper-ieee.pdf

[101] F. G. Frobenius. Über die Integration der linearen Differentialgleichungen durch Reihen. *Journal für die reine und angewandte Mathematik*, 76:214–235, 1873.
http://www.archive.org/details/journalfrdierei110crelgoog

[102] L. Fuchs. Zur Theorie der linearen Differentialgleichungen mit veränderlichen Coefficienten. *Journal für die reine und angewandte Mathematik*, 66:121–160, 1866.
http://www.archive.org/details/journalfrdierei63crelgoog

[103] K. Fukuda, J. van der Hoeven, M. Joswig et N. Takayama, éditeurs. *Mathematical Software – ICMS 2010*, volume 6327 de *Lecture Notes in Computer Science*. Springer, 2010.

[104] M. Fürer. Faster integer multiplication. *SIAM Journal on Computing*, 39(3):979–1005, 2009.
http://www.cse.psu.edu/~furer/Papers/mult.pdf

[105] K. Geddes. ALTRAN procedures for the Chebyshev series solution of linear ODE's. Research report CS-77-05, Department of Computer Science, University of Waterloo, 1977.
http://www.cs.uwaterloo.ca/research/tr/1977/CS-77-05.pdf

[106] K. Geddes. Symbolic computation of recurrence equations for the Chebyshev series solution of linear ODE's. In *Proceedings of the 1977 MACSYMA User's Conference* [1], pages 405–423.
http://dspace.udel.edu:8080/dspace/handle/19716/1398

[107] K. Geddes et J. Mason. Polynomial approximation by projections on the unit circle. *SIAM Journal on Numerical Analysis*, 12:111–120, 1975.

[108] J. Gerhard, M. Giesbrecht, A. Storjohann et E. V. Zima. Shiftless decomposition and polynomial-time rational summation. In *ISSAC '03* [219], pages 119–126.

[109] S. Gerhold. *Combinatorial Sequences: Non-Holonomicity and Inequalities*. PhD thesis, Johannes-Kepler-Universität Linz, 2005.
http://www.fam.tuwien.ac.at/~sgerhold/pub_files/diss.pdf

[110] Introduction to the gfun package. gfun v. 3.61 Help Pages, 1992–2011.
http://algo.inria.fr/libraries/papers/gfun.html

[111] Gmp nightly build tuneup tables. Consulté le 20 avril 2011.
http://gmplib.org/devel/MUL_FFT_THRESHOLD.html

[112] W. Gosper. Strip mining in the abandoned orefields of nineteenth century mathematics. In *Computers in Mathematics* [58], pages 261–284.

[113] K. Goto et R. A. van de Geijn. Anatomy of high-performance matrix multiplication. *ACM Transactions on Mathematical Software*, 34(3), 2008.
http://www.cs.utexas.edu/users/pingali/CS378/2008sp/papers/gotoPaper.pdf

[114] X. Gourdon et B. Salvy. Effective asymptotics of linear recurrences with rational coefficients. *Discrete Mathematics*, 153(1-3):145–163, 1996.

[115] X. Gourdon et P. Sebah. Binary splitting method, 2001.
http://numbers.computation.free.fr/Constants/Algorithms/splitting.ps

[116] T. Granlund et al. *GNU Multiple Precision Arithmetic Library*.
http://gmplib.org

[117] J. Gray. *Linear differential equations and group theory from Riemann to Poincaré*. Birkhäuser, 2000.

[118] D. A. Grier. Table making for the relief of labour. In M. Campbell-Kelly, M. Croarken, R. Flood et E. Robson, éditeurs, *The History of Mathematical Tables — From Sumer to Spreadsheets*, chapter 10, pages 265–294. Oxford University Press, 2003.

[119] M. Grimmer. Interval arithmetic in Maple with intpakX. *Proceedings in Applied Mathematics and Mecanics*, 2:442–443, 2003.
http://www2.math.uni-wuppertal.de/~xsc/software/intpakX/

[120] A. O. Guelfond. *Calcul des différences finies*. Collection Universitaire de Mathématiques, XII. Dunod, 1963. Traduit par G. Rideau.

[121] B. Haible et R. B. Kreckel. *CLN, a Class Library for Numbers*.
`http://www.ginac.de/CLN/cln.html`

[122] B. Haible et T. Papanikolaou. Fast multiprecision evaluation of series of rational numbers, 1997.
`http://www.informatik.tu-darmstadt.de/TI/Mitarbeiter/papanik/ps/TI-97-7.ps.gz`

[123] V. Halava, T. Harju, M. Hirvensalo et J. Karhumäki. Skolem's problem: On the border between decidability and undecidability. Technical Report 683, Turku Centre for Computer Science, 2005.
`http://www.tucs.fi/publications/attachment.php?fname=TR683.pdf`

[124] W. Hart, S. Pancratz, A. Novocin, F. Johansson, D. Harvey et al. Flint: Fast library for number theory, 2007–2011.
`http://flintlib.org/`

[125] M. Hazewinkel, éditeur. *Encyclopaedia of Mathematics*. Springer, 2002.
`http://eom.springer.de/`

[126] L. Heffter. *Einleitung in die Theorie der linearen Differentialgleichungen*. Teubner, 1894.
`http://www.archive.org/details/einleitungindie00heffgoog`

[127] P. Henrici. *Applied and Computational Complex Analysis*, volume II. Wiley-Interscience, 1977.

[128] E. Hille. *Ordinary differential equations in the complex domain*. Wiley, 1976. Dover reprint, 1997.

[129] J. Hoefkens. *Rigorous Numerical Analysis with High-Order Taylor Models*. PhD thesis, Michigan State University, 2001.
`http://bt.pa.msu.edu/pub/papers/hoefkensphd/hoefkensphd.pdf`

[130] G. K. Immink. *Asymptotics of Analytic Difference Equations*, volume 1085 de *Lecture Notes in Mathematics*. Springer, 1984.

[131] G. K. Immink. Reduction to canonical forms and the Stokes phenomenon in theory of linear difference equations. *SIAM Journal on Mathematical Analysis*, 22(1):238–259, 1991.

[132] E. L. Ince. *Ordinary Differential Equations*. Dover, 1956.

[133] ISO/IEC JTC 1/SC 22/WG 14. Programming languages — C. International standard ISO/IEC 9899, 1999. Second edition.
`http://www.open-std.org/jtc1/sc22/wg14/`

[134] E. Jeandel. Évaluation rapide de fonctions hypergéométriques. Rapport technique RT-0242, INRIA-ENS Lyon, 2000.
`http://www.inria.fr/rrrt/rt-0242.html`

[135] J. R. Johnson, H. Park et E. Kaltofen, éditeurs. *ISSAC '09: Proceedings of the 2009 international symposium on Symbolic and algebraic computation*. ACM, 2009.

[136] M. Joldeş. *Rigorous polynomial approximations and applications*. Thèse de doctorat, École normale supérieure de Lyon, 2011.
`http://perso.ens-lyon.fr/mioara.joldes/these/theseJoldes.pdf`

[137] R. Jungen. Sur les séries de Taylor n'ayant que des singularités algébrico-logarithmiques sur leur cercle de convergence. *Commentarii Mathematici Helvetici*, 3:266–306, 1931.

[138] E. A. Karatsuba. Fast evaluation of hypergeometric functions by FEE. In N. Papamichael, S. Ruscheweyh et E. B. Saff, éditeurs, *Proceedings of the 3rd CMFT conference on computational methods and function theory*, pages 303–314, 1999.

[139] E. W. Kaucher et W. L. Miranker. *Self-validating numerics for function space problems*. Academic Press, 1984.

[140] E. W. Kaucher et W. L. Miranker. Validating computation in a function space. In *Reliability in computing: the role of interval methods in scientific computing*, pages 403–425. Academic Press Professional, Inc., 1988.

[141] M. Kauers. Computer algebra for special function inequalities. *Contemporary Mathematics*, 457:215–236, 2008.
`http://www.risc.uni-linz.ac.at/people/mkauers/publications/kauers07q.pdf`

[142] M. Kauers. A Mathematica package for computing asymptotic expansions of solutions of P-finite recurrence equations. Technical Report RISC 11-04, Johannes Kepler Universität Linz, avril 2011.
`http://www.risc.uni-linz.ac.at/people/mkauers/publications/kauers11e.pdf`

[143] M. Kauers et V. Pillwein. When can we detect that a P-finite sequence is positive? In *ISSAC '10* [148], pages 195–201.
`http://arxiv.org/abs/1005.0600`

[144] D. E. Knuth. Big omicron and big omega and big theta. *ACM Sigact News*, 8(2):18–24, 1976.

[145] D. E. Knuth. *Seminumerical Algorithms*, volume 2 de *The Art of Computer Programming*. Addison-Wesley, third edition, 1997.

[146] D. E. Knuth. *Sorting and Searching*, volume 3 de *The Art of Computer Programming*. Addison-Wesley, second edition, 1998.

[147] W. Koepf. Algebraische Darstellung transzendenter Funktionen. *Computeralgebra-Rundbrief*, 16:12–18, 1995.
`http://arxiv.org/abs/math/9412226`

[148] W. Koepf, éditeur. *ISSAC '10: Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation*. ACM, 2010.

[149] C. Koutschan. *Advanced Applications of the Holonomic Systems Approach*. PhD thesis, Johannes-Kepler-Universität Linz, 2009.
http://www.risc.jku.at/publications/download/risc_3886/thesisKoutschan.pdf

[150] R. B. Kreckel. decimal(γ) =∼ "0.57721566[0-9]{1001262760}39288477", janvier 2008.
http://www.ginac.de/~kreckel/news.html

[151] P. Kreuser. *Über das Verhalten der Integrale homogener linearer Differenzengleichungen im Unendlichen*. PhD thesis, Universität Tübingen, Borna-Leipzig, 1914.

[152] E. L. Lafferty. Power series solutions of ordinary differential equations in MACSYMA. In *Proceedings of the 1977 MACSYMA User's Conference* [1], pages 347–360.
http://dspace.udel.edu:8080/dspace/handle/19716/1398

[153] C. Lanczos. Trigonometric interpolation of empirical and analytical functions. *Journal of Mathematical Physics*, 17:123–199, 1938.

[154] C. Lanczos. *Applied analysis*. Prentice-Hall, 1956.

[155] C. Q. Lauter. *Arrondi correct de fonctions mathématiques – Fonctions univariées et bivariées, certification et automatisation*. Thèse de doctorat, Université de Lyon – École Normale Supérieure de Lyon, 2008.
http://www.ens-lyon.fr/LIP/Pub/Rapports/PhD/PhD2008/PhD2008-07.pdf

[156] B. Levin. *Lectures on entire functions*. American Mathematical Society, 1996.

[157] S. Lewanowicz. Construction of a recurrence relation of the lowest order for coefficients of the Gegenbauer series. *Zastosowania Matematyki*, XV(3):345–395, 1976.

[158] S. Lewanowicz. A new approach to the problem of constructing recurrence relations for the Jacobi coefficients. *Zastosowania Matematyki*, 21:303–326, 1991.

[159] L. Lipshitz. *D*-finite power series. *Journal of Algebra*, 122(2):353–373, 1989.

[160] R. J. Lipton. Galactic algorithms, octobre 2010. Consulté le 26 avril 2011.
http://rjlipton.wordpress.com/2010/10/23/galactic-algorithms/

[161] M. Loday-Richaud. Communication personnelle, janvier 2012.

[162] P. Luschny. Fast-factorial-functions: The homepage of factorial algorithms, 2000-2010. Consulté le 18 mai 2011.
http://luschny.de/math/factorial/FastFactorialFunctions.htm

[163] O. M. Makarov. An algorithm for multiplying 3×3 matrices. *USSR Computational Mathematics and Mathematical Physics*, 26(1):179–180, 1987. Traduit de Zhurnal Vychislitelnoi Matematiki I Matematicheskoi Fiziki, 26(2):293-294, 1986.

[164] K. Makino et M. Berz. Taylor models and other validated functional inclusion methods. *International Journal of Pure and Applied Mathematics*, 4(4):379–456, 2003.
http://bt.pa.msu.edu/pub/papers/TMIJPAM03/TMIJPAM03.pdf

[165] C. Mallinger. Algorithmic manipulation and transformations of univariate holonomic functions and sequences. Diplomarbeit, RISC-Linz, 1996.
http://www.risc.jku.at/publications/download/risc_2244/DIPLFORM.pdf

[166] Maplesoft. Maple, 1982-2011.
http://www.maplesoft.com/products/maple/

[167] Maplesoft. DEtools[formal_sol] — formal solutions of a homogeneous linear ode. In *Maple 15 Online Help*. Maplesoft, 2011.
http://www.maplesoft.com/support/help/Maple/view.aspx?path=formal_sol

[168] Maplesoft. The five Second Order Linear Heun equations and the corresponding Heun function solutions. In *Maple 15 Online Help*. Maplesoft, 2011.
http://www.maplesoft.com/support/help/Maple/view.aspx?path=Heun

[169] J. Mason et D. Handscomb. *Chebyshev polynomials*. CRC Press, 2003.

[170] R. J. Mathar. Chebyshev series expansion of inverse polynomials. *Journal of Computational and Applied Mathematics*, 196(2):596–607, 2006.
http://arxiv.org/abs/math/0403344

[171] H. Meschkowski. *Differenzengleichungen*. Vandenhoeck & Ruprecht, 1959.

[172] L. Meunier et B. Salvy. ESF: an automatically generated encyclopedia of special functions. In *ISSAC '03* [219], pages 199–206.
http://algo.inria.fr/meunier/articles/MeSa03.pdf

[173] M. Mezzarobba. Génération automatique de procédures numériques pour les fonctions D-finies. Rapport de stage, Master parisien de recherche en informatique, octobre 2007. Version 1.2.
http://marc.mezzarobba.net/m2/Mezzarobba_MScThesisMPRI2007-1.2.pdf

[174] M. Mezzarobba. NumGfun: a package for numerical and analytic computation with D-finite functions. In *ISSAC '10* [148], pages 139–146.
http://arxiv.org/abs/1002.3077

[175] M. Mezzarobba et B. Salvy. Effective bounds for P-recursive sequences. *Journal of Symbolic Computation*, 45(10):1075–1096, 2010.

http://arxiv.org/abs/0904.2452

[176] M. Mezzino et M. Pinsky. Leibniz's formula, Cauchy majorants, and linear differential equations. *Mathematics Magazine*, 71(5):360–368, 1998.

[177] L. Milne-Thomson. *The calculus of finite differences*. Macmillan, 1933.

[178] M. Monagan. Maximal quotient rational reconstruction: an almost optimal algorithm for rational reconstruction. In J. Gutierrez, éditeur, *ISSAC '04*, pages 243–249. ACM, 2004.
http://ftp.cecm.sfu.ca/personal/monaganm/papers/MQIRR.pdf

[179] R. Moore. *Methods and applications of interval analysis*. Society for Industrial and Applied Mathematics, 1979.

[180] R. Moore, R. Kearfott et M. Cloud. *Introduction to interval analysis*. Society for Industrial and Applied Mathematics, 2009.

[181] R. E. Moore. *Interval arithmetic and automatic error analysis in digital computing*. PhD thesis, Stanford University, 1962. Published as Applied Mathematics and Statistics Laboratories Technical Report No. 25.

[182] J. Moser. The order of a singularity in Fuch's theory. *Mathematische Zeitschrift*, 72:379–398, 1960.

[183] MPFR Team. The GNU MPFR library, 2000-2011.
http://www.mpfr.org/

[184] MPIR Team. MPIR: Multiple precision integers and rationals, 2009–2011. MPIR est une bibliothèque dérivée de GMP [116].
http://www.mpir.org/

[185] J.-M. Muller. *Elementary functions: algorithms and implementation*. Birkhäuser, 1997.

[186] J.-M. Muller, N. Brisebarre, F. de Dinechin, C.-P. Jeannerod, V. Lefèvre, G. Melquiond, N. Revol, D. Stehlé et S. Torres. *Handbook of Floating-Point Arithmetic*. Birkhäuser, 2010.

[187] G. M. Murphy. *Ordinary Differential Equations and Their Solutions*. Van Nostrand, 1960.

[188] N. Möller. Re: fast gcd. Message sur la liste de diffusion gmp-devel, octobre 2009.
http://gmplib.org/list-archives/gmp-devel/2009-October/001035.html

[189] M. Neher. An enclosure method for the solution of linear ODEs with polynomial coefficients. *Numerical Functional Analysis and Optimization*, 20:779–803, 1999.
http://iamlasun8.mathematik.uni-karlsruhe.de/~ae16/preprnts/neher_1999_polynomial_ivp_NFAO20.pdf

[190] M. Neher. Improved validated bounds for Taylor coefficients and for Taylor remainder series. *Journal of Computational and Applied Mathematics*, 152:393–404, 2003.
http://iamlasun8.mathematik.uni-karlsruhe.de/~ae16/preprnts/neher_2003_improved_taylor_JCAM152.pdf

[191] M. Neher, K. R. Jackson et N. S. Nedialkov. On Taylor model based integration of ODEs. *SIAM Journal on Numerical Analysis*, 45(1):236–262, 2007.
http://iamlasun8.mathematik.uni-karlsruhe.de/~ae16/preprnts/neher_jackson_nedialkov_2005_TM_ODEs_uka0521.pdf

[192] A. Neumaier. Taylor forms – Use and limits. *Reliable Computing*, 9(1):43–79, 2003.
http://www.mat.univie.ac.at/~neum/ms/taylor.pdf

[193] R. Noble. *Zeros and Asymptotics of Holonomic Sequences*. PhD thesis, Dalhousie University, Halifax, Nova Scotia, mars 2011.
http://hdl.handle.net/10222/13298

[194] A. M. Odlyzko. Asymptotic enumeration methods. In R. L. Graham, M. Groetschel et L. Lovasz, éditeurs, *Handbook of Combinatorics*, volume 2, pages 1063–1229. Elsevier, 1995.
http://www.dtc.umn.edu/~odlyzko/doc/asymptotic.enum.pdf

[195] F. Olver, D. Lozier, R. Boisvert et C. Clark, éditeurs. *NIST Handbook of Mathematical Functions*. Cambridge University Press, 2010.

[196] V. Y. Pan. Optimal and nearly optimal algorithms for approximating polynomial zeros. *Computers & Mathematics with Applications*, 31(12):97–138, 1996.
http://comet.lehman.cuny.edu/vpan/pdf/pan129.pdf

[197] V. Y. Pan. New fast algorithms for polynomial interpolation and evaluation on the Chebyshev node set. *Computers & Mathematics with Applications*, 35(3):125–129, 1998.
http://comet.lehman.cuny.edu/vpan/fromVAX/chebyshev98_150.tex

[198] S. Paszkowski. Zastosowania numeryczne wielomianow i szeregow Czebyszewa. *Podstawowe Algorytmy Numeryczne*, 1975.

[199] O. Perron. Über lineare Differenzengleichungen. *Acta Mathematica*, 34:109–137, 1910.
http://www.archive.org/details/actamathematica34upps

[200] O. Perron. Über lineare Differenzengleichungen und eine Anwendung auf lineare Differentialgleichungen mit Polynomkoeffizienten. *Mathematische Zeitschrift*, 72(1):16–24, 1959.

[201] M. Petkovšek. Hypergeometric solutions of linear recurrences with polynomial coefficients. *Journal of Symbolic Computation*, 14(2-3):243–264, 1992.
http://www.fmf.uni-lj.si/~petkovsek/hyp.PS

[202] H. Poincaré. Sur les équations linéaires aux différentielles ordinaires et aux différences finies. *American Journal of Mathematics*, 7(3):203–258, 1885.

[203] E. G. C. Poole. *Introduction to the theory of linear differential equations*. Clarendon Press, 1936.

[204] A. Poteaux. *Calcul de développements de Puiseux et application au calcul du groupe de monodromie d'une courbe algébrique plane*. Thèse de doctorat, Université de Limoges, 2008.
`http://epublications.unilim.fr/theses/2008/poteaux-adrien/poteaux-adrien.pdf`

[205] Projet Algorithms. Algolib, version 14.0, décembre 2010.
`http://algo.inria.fr/libraries/`

[206] A. P. Prudnikov, Y. A. Brychkov et O. I. Marichev. *Integrals and Series. Volume 1: Elementary functions*. Gordon and Breach, 1986.

[207] L. Rebillard. *Etude théorique et algorithmique des séries de Chebyshev solutions d'équations différentielles holonomes*. Thèse de doctorat, Institut national polytechnique de Grenoble, 1998.
`http://tel.archives-ouvertes.fr/tel-00008571/`

[208] D. Richardson. Some undecidable problems involving elementary functions of a real variable. *Journal of Symbolic Logic*, 33(4):514–520, 1968.

[209] R. Rihm. Interval methods for initial value problems in ODEs. In J. Herzberger, éditeur, *Topics in Validated Computations: Proceedings of the IMACS-GAMM International Workshop on Validated Computations, University of Oldenburg*, Elsevier Studies in Computational Mathematics, pages 173–207. Elsevier, 1994.

[210] T. J. Rivlin. *The Chebyshev polynomials*. Wiley, 1974.

[211] P. Rothfuss. *The Name of the Wind*. DAW Books, 2008.

[212] E. Salamin. Computation of π using arithmetic-geometric mean. *Mathematics of Computation*, 30(135):565–570, 1976.

[213] B. Salvy. Examples of automatic asymptotic expansions. *SIGSAM Bulletin*, 25(2):4–17, avril 1991.
`http://hal.archives-ouvertes.fr/inria-00070052`

[214] B. Salvy et P. Zimmermann. Gfun: A Maple package for the manipulation of generating and holonomic functions in one variable. *ACM Transactions on Mathematical Software*, 20(2):163–177, 1994.

[215] W. Schlag. A concise course in complex analysis and Riemann surfaces. Lecture notes.
`http://www.math.uchicago.edu/~schlag/complex.pdf`

[216] F. W. Schäfke. Lösungstypen von Differenzengleichungen und Summengleichungen in normierten abelschen Gruppen. *Mathematische Zeitschrift*, 88(1):61–104, février 1965.

[217] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity. Technical report, Mathematisches Institut der Universität Tübingen, 1982.
`http://www.iai.uni-bonn.de/~schoe/fdthmrep.ps.gz`

[218] A. Schönhage, A. F. W. Grotefeld et E. Vetter. *Fast algorithms – A multitape Turing machine implementation*. B.I. Wissenschaftsverlag, 1994.
`http://www.iai.uni-bonn.de/~schoe/tp/TPbook.html`

[219] J. R. Sendra, éditeur. *ISSAC '03: Proceedings of the 2003 international symposium on Symbolic and algebraic computation*. ACM, 2003.

[220] N. J. A. Sloane et al. The on-line encyclopedia of integer sequences (OEIS), 1996–2011.
`http://oeis.org/`

[221] E. Solomentsev. Riemann surface. In *Encyclopaedia of Mathematics* [125].
`http://eom.springer.de/`

[222] R. P. Stanley. Differentiably finite power series. *European Journal of Combinatorics*, 1(2):175–188, 1980.

[223] R. P. Stanley. *Enumerative combinatorics*, volume 2. Cambridge University Press, 1999.

[224] A. Steel. Reduce everything to multiplication. Slides of a talk at the workshop "Computing by the Numbers: Algorithms, Precision, and Complexity" for R. Brent 60th birthday, juillet 2006.
`http://www.mathematik.hu-berlin.de/~gaggle/EVENTS/2006/BRENT60/presentations/Allan Steel - Reduce everything to multiplication.pdf`

[225] W. Stein et al. Sage: Open source mathematics software, 2005–2011.
`http://sagemath.org/`

[226] A. Storjohann. Faster algorithms for integer lattice basis reduction. Technical Report 249, Eidgenössische Technische Hochschule Zürich, Department of Computer Science, 1996.
`ftp://ftp.inf.ethz.ch/doc/tech-reports/2xx/249.ps.gz`

[227] A. Storjohann. The shifted number system for fast linear algebra on integer matrices. *Journal of Complexity*, 21(4):609–650, 2005.

[228] A. Storjohann. On the complexity of inverting integer and polynomial matrices. Accepted for publication in Computational Complexity, décembre 2008.
`http://www.cs.uwaterloo.ca/~astorjoh/geninv.pdf`

[229] L. Swann. *Glennkill*. Goldmann, 2007.

[230] É. Tournier. *Solutions formelles d'équations différentielles*. Doctorat d'État, Université scientifique, technologique et médicale de Grenoble, 1987.
http://tel.archives-ouvertes.fr/tel-00323706/fr/

[231] L. N. Trefethen. Computing numerically with functions instead of numbers. *Mathematics in Computer Science*, 1(1):9–19, 2007.
http://people.maths.ox.ac.uk/trefethen/publication/PDF/2007_123.pdf

[232] L. N. Trefethen. *Approximation theory and approximation practice*. juin 2011.
http://www2.maths.ox.ac.uk/chebfun/ATAP/

[233] H. Tsai. Weyl closure of a linear differential operator. *Journal of Symbolic Computation*, 29(4-5):747–775, 2000.

[234] J. van der Hoeven. *Automatic asymptotics*. Thèse de doctorat, École polytechnique, 1997.
http://www.texmacs.org/joris/phd/phd-abs.html

[235] J. van der Hoeven. Fast evaluation of holonomic functions. *Theoretical Computer Science*, 210(1):199–216, 1999.
http://www.texmacs.org/joris/hol/hol-abs.html

[236] J. van der Hoeven. Fast evaluation of holonomic functions near and in regular singularities. *Journal of Symbolic Computation*, 31(6):717–743, 2001.
http://www.texmacs.org/joris/singhol/singhol-abs.html

[237] J. van der Hoeven. Majorants for formal power series. Technical Report 2003-15, Université Paris-Sud, Orsay, France, 2003.
http://www.texmacs.org/joris/maj/maj-abs.html

[238] J. van der Hoeven. Around the numeric-symbolic computation of differential Galois groups. *Journal of Symbolic Computation*, 42(1-2):236–264, 2007.
http://www.texmacs.org/joris/galois/galois-abs.html

[239] J. van der Hoeven. Efficient accelero-summation of holonomic functions. *Journal of Symbolic Computation*, 42(4):389–428, 2007.
http://www.texmacs.org/joris/reshol/reshol-abs.html

[240] J. van der Hoeven. On effective analytic continuation. *Mathematics in Computer Science*, 1(1):111–175, 2007.
http://www.texmacs.org/joris/riemann/riemann-abs.html

[241] J. van der Hoeven. *Transséries et analyse complexe effective*. Mémoire d'habilitation, Université Paris-Sud, Orsay, France, 2007.
http://www.texmacs.org/joris/hab/hab-abs.html

[242] J. van der Hoeven, G. Lecerf, B. Mourain, P. Trébuchet, J. Berthomieu, D. Diatta et A. Mantzaflaris. Mathemagix, the quest of modularity and efficiency for symbolic and certified numeric computation. *ACM SIGSAM Communications in Computer Algebra*, 177(3), 2011. ISSAC 2011 Software Demonstrations, edited by M. Stillman.
http://www.sigsam.org/bulletin/articles/177/CCA-177-full.pdf

[243] J. van der Hoeven, G. Lecerf, B. Mourrain et al. Mathemagix, 2001–2011.
http://www.mathemagix.org/

[244] M. van der Put et M. F. Singer. *Galois theory of difference equations*, volume 1666 de *Lecture Notes in Mathematics*. Springer, 1997.

[245] M. van der Put et M. F. Singer. *Galois Theory of Linear Differential Equations*, volume 328 de *Grundlehren der mathematischen Wissenschaften*. Springer, 2003.
http://www4.ncsu.edu/~singer/papers/dbook2.ps

[246] M. van Hoeij. My Maple code.
http://www.math.fsu.edu/~hoeij/maple.html

[247] M. van Hoeij. Formal solutions and factorization of differential operators with power series coefficients. *Journal of Symbolic Computation*, 24(1):1–30, 1997.
http://www.math.fsu.edu/~hoeij/papers/comments/jsc1997b.html

[248] G. Villard. *Algorithmique en algèbre linéaire exacte*. Mémoire d'habilitation, Université Claude Bernard Lyon 1, mars 2003.
http://perso.ens-lyon.fr/gilles.villard/BIBLIOGRAPHIE/PDF/Hdr.pdf

[249] J. von zur Gathen et J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2nd edition, 2003.

[250] A. Waksman. On Winograd's algorithm for inner products. *IEEE Transactions on Computers*, C-19(4):360–361, 1970.

[251] X. Wang et V. Pan. Acceleration of Euclidean algorithm and rational number reconstruction. *SIAM Journal on Computing*, 32(2):548, 2003.
http://comet.lehman.cuny.edu/vpan/pdf/40863.pdf

[252] W. Wasow. *Asymptotic expansions for ordinary differential equations*. Wiley, 1965. Dover reprint, 1987.

[253] S. Watanabe. Formula manipulations solving linear ordinary differential equations (I). *Publications of the Research Institute for Mathematical Sciences*, 6:71–111, 1970.

[254] R. C. Whaley et A. Petitet. Minimizing development and maintenance costs in supporting persistently optimized BLAS. *Software: Practice and Experience*, 35(2):101–121, février 2005.
http://www.cs.utsa.edu/~whaley/papers/spercw04.ps

[255] Wikipedia. Nachbin's theorem — Wikipedia, the free encyclopedia, 2011. Version du 21 avril 2011 (id=414069739).
http://en.wikipedia.org/wiki/Nachbin's_theorem?oldid=414069739

[256] J. Wimp. *Computation with Recurrence Relations*. Pitman, 1984.

[257] J. Wimp. Current trends in asymptotics: some problems and some solutions. *Journal of Computational and Applied Mathematics*, 35(1-3):53–79, 1991.

[258] J. Wimp et D. Zeilberger. Resurrecting the asymptotics of linear recurrences. *Journal of Mathematical Analysis and Applications*, 111:162–176, 1985.
http://www.math.rutgers.edu/~zeilberg/mamarimY/Zeilberger_y1985_p162.pdf

[259] Wolfram Research. Mathematica, 1988-2011.
http://www.wolfram.com/products/mathematica/

[260] Yacas Team. *The Yacas Book of Algorithms*, Yacas v. 1.0.63 edition, janvier 2007.
http://yacas.sourceforge.net/Algo.book.pdf

[261] S. V. Yakhontov. A simple algorithm for the evaluation of the hypergeometric series using quasi-linear time and linear space. Preprint 1106.2301v1, arXiv, juin 2011.
http://arxiv.org/abs/1106.2301

[262] A. J. Yee et S. Kondo. 5 trillion digits of pi — new world record, 2010. Version mise à jour le 7 mars 2011, consultée le 24 avril 2011.
http://www.numberworld.org/misc_runs/pi-5t/details.html

[263] R. V. Zahar. A mathematical analysis of Miller's algorithm. *Numerische Mathematik*, 27(4):427–447, 1976.

[264] D. Zeilberger. A holonomic systems approach to special functions identities. *Journal of Computational and Applied Mathematics*, 32(3):321–368, 1990.
http://www.math.rutgers.edu/~zeilberg/mamarim/mamarimhtml/holonomic.html

[265] D. Zeilberger. The method of creative telescoping. *Journal of Symbolic Computation*, 11:195–204, 1991.
http://www.math.rutgers.edu/~zeilberg/mamarimY/Zeilberger_y1991_p195.pdf

[266] D. Zeilberger. Asyrec: A Maple package for computing the asymptotics of solutions of linear recurrence equations with polynomial coefficients, apr 2008.
http://www.math.rutgers.edu/~zeilberg/mamarim/mamarimhtml/asy.html

[267] P. Zimmermann. The bit-burst algorithm. Slides of a talk at the workshop "Computing by the Numbers: Algorithms, Precision, and Complexity" for R. Brent 60th birthday, juillet 2006.
http://www.mathematik.hu-berlin.de/~gaggle/EVENTS/2006/BRENT60/presentations/Paul Zimmermann - The bit-burst algorithm.pdf

Index