UNIVERSITÉ DE LIMOGES
ÉCOLE DOCTORALE Science – Technologie – Santé
FACULTÉ des sciences et techniques

Laboratoire d'**A**rithmétique, de **C**alcul formel et d'**O**ptimisation

Thèse

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LIMOGES

Discipline : Mathématiques Spécialité : Calcul formel

présentée et soutenue

par

Thomas CLUZEAU

le 23 septembre 2004

Algorithmique modulaire des équations différentielles linéaires

Thèse dirigée par Moulay A. Barkatou et Jacques-Arthur Weil

Jury

Président	Jean-Pierre Ramis, Professeur à l'université de Toulouse
Rapporteurs	Anne Duval, Professeur à l'université de Lille Mark Giesbrecht, Professeur à l'université de Waterloo (Canada)
Examinateurs	Manuel Bronstein, Directeur de recherche à l'INRIA Mark van Hoeij, Professeur à l'université d'État de Floride (É.-U.) Bruno Salvy, Directeur de recherche à l'INRIA
Directeurs	Moulay A. Barkatou, Professeur à l'université de Limoges Jacques-Arthur Weil, Maître de conférences à l'université de Limoges

UNIVERSITÉ DE LIMOGES
ÉCOLE DOCTORALE Science – Technologie – Santé
FACULTÉ des sciences et techniques

Laboratoire d'**A**rithmétique, de **C**alcul formel et d'**O**ptimisation

Thèse

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LIMOGES

Discipline : Mathématiques Spécialité : Calcul formel

présentée et soutenue

par

Thomas CLUZEAU

le 23 septembre 2004

Algorithmique modulaire des équations différentielles linéaires

Thèse dirigée par Moulay A. Barkatou et Jacques-Arthur Weil

Jury

Président	Jean-Pierre Ramis, Professeur à l'université de Toulouse
Rapporteurs	Anne Duval, Professeur à l'université de Lille Mark Giesbrecht, Professeur à l'université de Waterloo (Canada)
Examinateurs	Manuel Bronstein, Directeur de recherche à l'INRIA Mark van Hoeij, Professeur à l'université d'État de Floride (É.-U.) Bruno Salvy, Directeur de recherche à l'INRIA
Directeurs	Moulay A. Barkatou, Professeur à l'université de Limoges Jacques-Arthur Weil, Maître de conférences à l'université de Limoges

Remerciements

Cette thèse a été préparée entre 2000 et 2004 au sein du Laboratoire d'Arithmétique, Calcul formel et Optimisation (LACO) de l'université de Limoges et a été financée, de 2000 à 2003, par la région Limousin.

De nombreuses personnes ont participé, plus ou moins directement, à la réussite de cette thèse : je vais essayer de vous les présenter en espérant n'oublier personne.

Pourquoi une thèse en calcul formel ? En effectuant un retour quatre ans en arrière, je m'aperçois que trois personnes ont contribué directement au fait que je me sois lancé dans une thèse en calcul formel : tout d'abord, au cours de ma maîtrise à Limoges, j'ai été fasciné par la manière avec laquelle Jacques-Arthur Weil conduisait son cours de calcul formel ; malgré des horaires matinaux, il réussissait toujours à rendre le cours vivant et ce grâce à un enthousiasme hors du commun. Ensuite, durant mon DEA, j'ai eu la chance de suivre un cours de Jean-Claude Yakoubsohn. Là encore, j'ai senti que plus que nous transmettre des connaissances, l'orateur nous faisait partager sa passion. Enfin, Évelyne Hubert a dirigé mon stage de DEA et donc, en quelque sorte, guidé mes premiers pas dans la recherche. Ce stage a certainement servi de déclic au départ de cette aventure ; elle y est donc pour beaucoup. Ces trois personnes sont à l'origine de cette thèse et je leur en suis donc très reconnaissant.

Jacques-Arthur Weil m'a proposé un sujet de thèse ouvert et passionnant. Tout au long de ce travail, ses encouragements, son enthousiasme débordant et sa curiosité scientifique ont toujours réussi à stimuler ma motivation. Il a su diriger ce travail même lors des deux années qu'il a passé à Nice et nos conversations, scientifiques ou non, ont été une aide précieuse pour moi. Qu'il accepte tous mes remerciements et ma reconnaissance.

Cette thèse était déjà commencée depuis un an lorsque Moulay Barkatou est arrivé à Limoges. Cependant, grâce à ses larges connaissances mathématiques, il a su immédiatement proposer des directions de recherche qui se sont avérées fructueuses. Sa rigueur scientifique m'a toujours aidé à clarifier les problèmes. Elle a aussi permis de canaliser les « délires utopiques » de Jacques-Arthur. Il n'a cessé de répondre, avec entrain, à mes questions et ce, dans un climat des plus convivial. Qu'il trouve ici l'expression de ma gratitude.

Mark van Hoeij peut, en quelque sorte, être considéré comme mon troisième directeur de thèse. Il m'a accueilli deux mois chez lui à Tallahassee (en Floride) et a ainsi pu me faire profiter de toutes ses connaissances et me communiquer sa passion pour la recherche. Il a co-signé avec moi deux articles qui constituent les chapitres deux et trois de cette thèse. Sans lui, ce travail n'existerait pas et c'est donc un immense plaisir pour moi qu'il participe à mon jury. Je profite de l'occasion pour remercier aussi sa femme Susan pour l'accueil formidable qu'elle m'a réservé lors de mes deux séjours à

Tallahassee.

Depuis mon stage de DEA, j'ai eu la chance d'être invité plusieurs fois au projet CAFE de l'INRIA de Sophia Antipolis. Lors de ces séjours, Manuel Bronstein m'a réservé un accueil chaleureux. De plus, il a toujours montré beaucoup d'intérêt pour mon travail et m'a, à de nombreuses reprises, donné des conseils très pertinents. Je tiens à l'en remercier et je suis très honoré qu'il ai accepté de participer à mon jury.

L'année que j'ai passé au laboratoire STIX de l'École polytechnique m'a permis de cotoyer Bruno Salvy. Lors des séances de travail que nous avons pu partager, il a toujours répondu, avec entrain et dans un climat très convivial, aux nombreuses questions (parfois très naïves) que j'ai pu lui poser. Je suis heureux qu'il ai bien voulu participer à mon jury.

Je suis très reconnaissant à Anne Duval et Mark Giesbrecht d'avoir accepté d'être rapporteurs ; je les remercie d'avoir lu, en détails, cette thèse et d'avoir ainsi témoigné de l'attention à mes travaux. Leurs remarques judicieuses ont permis d'améliorer la qualité de ce mémoire.

C'est un immense honneur pour moi que Jean-Pierre Ramis ai accepté de prendre part à mon jury et qu'il ai ainsi montré de l'intérêt à mon travail. Je lui en suis très reconnaissant.

Les articles de Marius van der Put ont servi de point de départ à mon travail. Au cours de ma thèse, il m'a donné certaines directions de recherche très pertinentes. Qu'il en soit remercié.

L'environnement de travail est un ingrédient indispensable à la réussite d'une thèse. J'ai passé mes trois premières années dans le « bureau vert » en compagnie de Matthieu Lefloc'h et Mikaël Lescop. Il y a toujours reigné une ambiance très détendue, parfois peut être même trop en certaines périodes où la non-motivation de l'un des trois se propageait chez les autres. Même si nous avions rarement le même point de vue sur les actualités du monde du football ou du tennis, nous nous retrouvions toujours sur la même longueur d'onde à l'approche de l'été où le bureau devenait le siège du fan-club de Jan Ullrich. Je les remercie donc tous les deux pour avoir fait que cette thèse se déroule dans les meilleures conditions et je leur souhaite bon courage pour l'avenir. Je souhaite aussi bon courage au nouvel habitant du bureau Guilhem Castagnos, qui rentre parfaitement dans l'esprit « bureau vert ».

J'ai cohabité pendant plus de trois ans avec les membres du LACO. Je tiens à remercier les anciens et actuels thésards (Ma « grande soeur de recherche » Delphine Boucher, Cyril Brunie, Laurent Dubreuil, Meriem Heraoua, Nicolas Le Roux, Samuel Maffre, Carmen Nedeloaia, Ayoub Otmani, Philippe Ségalat et Cyril Vervoux) pour leur soutien permanent, tous les membres de l'équipe calcul formel ainsi qu'Alexandre Ca-

bot, Philippe Gaborit (pour ses conseils souvent pertinents), Henri Massias, Abdelkader Necer et Stéphane Vinatier. Je remercie tout particulièrement Yolande Vieceli dont la disponibilité et la bonne humeur contribuent pleinement à l'ambiance joyeuse qui règne dans les couloirs du bâtiment de mathématiques : que deviendrait le LACO sans elle ? Je remercie enfin Martine Guerletin, Sylvie Laval, Nadine Tchéfranoff et Patricia Vareille pour leur sympathie et la manière avec laquelle elles ont contribué à me faciliter les tâches administratives.

J'ai passé ma quatrième année de thèse au sein du laboratoire STIX de l'École polytechnique. Je remercie Jean Moulin-Ollagnier qui a soutenu mes candidatures sur les postes d'ATER à l'université de Paris XII ainsi que Marc Giusti qui m'a accueilli dans son laboratoire et m'a ainsi fait bénéficier de conditions de travail idéales. Je remercie Xavier Dahan, Anne Fredet et Pierre Midy qui ont été mes collègues de bureau pendant toute cette année. Mention spéciale à ma « deuxième grande soeur de recherche » Anne Fredet avec qui travailler est toujours un plaisir et ce malgré nos divergences de point de vue sur de nombreux sujets (en fait presque sur tout !). Une grande partie du quatrième chapitre est un travail réalisé en collaboration avec elle et elle a donc fortement contribué à la réussite de cette thèse. Je remercie aussi Alin Bostan pour s'être intéressé à mon travail depuis le début et avec qui il est toujours très agréable de travailler. Je n'oublie pas tous les autres membres du laboratoire STIX et tout particulièrement Kilian Cavalotti, Nicole Dubois, Christian Fleck, Pierre Lafon (peut être mon futur collègue de location de pédalos aux Bahamas), Grégoire Lecerf, François Ollivier, Éric Schost et Bernd Wiebelt.

Je remercie aussi Jean-Baptiste Aubin et Édouard Pennamen avec qui ce fût un grand plaisir de travailler à Paris XII ainsi que Jean-Marie Aubry pour m'avoir permis d'enseigner des matières tout à fait intéressantes.

Durant ma thèse, j'ai été amené à beaucoup voyager, soit pour assister à des conférences, soit pour participer à des séminaires. Cela m'a permis de rencontrer plusieurs personnes que je souhaite remercier : Alban Quadrat, Raphaël Bomboy, Gwénolé Ars, Olivier Cormier, Solen Corvez, Félix Ulmer, Philippe Aubry, Magali Bardet, Mohab Safey El Din, Guénaël Renault, Philippe Trébuchet, Olivier Ruatta, Guillaume Chèze, Frédéric Chyzak, Hà Lê, Ludovic Meunier (merci pour la visite guidée des environs d'un hôpital de Philadelphie), A. H. M. Levelt, Élie Compoint (à qui j'ai réussi à faire apprécier un disque de heavy metal), B. Heinrich Matzat, Julia Hartmann, Joris van der Hoeven, Georges Labahn, Claude-Pierre Jeannerod, Jean-Guillaume Dumas, Yang Zhang et Sergei A. Abramov.

Tout au long de cette thèse, j'ai eu la chance de pouvoir toujours compter sur ma famille et mes amis. Même si leur contribution à ce travail paraît moindre, je suis persuadé qu'en m'accompagnant dans la vie de tous les jours, ils m'ont permis de mener cette thèse à termes dans les meilleures conditions. Je remercie donc chaleureusement mes parents, mon frère et Geneviève ainsi que toute ma famille, mon « brother of metal »

Franck ainsi que sa femme Sophie, sans oublier Marie, Valentine et Victor, Pierre (dit « le tocard »), Karine, Anne, David et toute la famille Aubreton, sans oublier Sophie, Richard et la famille Auroux. J'ai une pensée pour tous mes amis du Tennis Club de Feytiat (certains étant aussi d'excellents camarades de soirées !) : Ben « Poupoune » Fredon, Pierre Miran et sa femme Sandrine (c'est bon ça !), Bruno Bernard-Coffre, David Petitet, Olivier Carpe (alias Tony Montana), Xavier Chanut et Laurence Dartois (aussi connue sous les pseudos de Kim ou Abby), ainsi que pour tous ceux avec qui j'ai eu la chance de partager d'autres grandes soirées : Éric, Carole, Pascal et la famille Beyssac, Charles, Alexandra, Hugues et la famille François, Arnaud « Nono » Boscher et le grand « Gourou » Emmanuel Cadic (pour les soirées de méditation au temple), François Delost et Maude Guillot (entre autre pour les soirées dégustation de petite poire ... ça va me manquer !), les deux autres membres de la couinche ligue, *i. e.*, Gilles Reynaud et Xavier Sidot, Steph' et Guillaume de l'O'brien Tavern pour leur accueil toujours formidable, Sandra pour sa folie de tous les instants, Delphine, Christophe, Jean-Marc, Matmaz, toute la grande troupe du water-polo limougeaud (les filles, les gars sans oublier Jérôme et Charles), PSG (aussi surnommé Jean-Pierre Bacri ou POB pour Philippe of O'Brien), Christian Vinçon, Sylvain Jeamot et Marc de Roos, mes amis physiciens (personne n'est parfait) Benoît Brousse et Matthieu Valetas et toute autre personne avec qui j'ai pu passer des bons moments.

Enfin je remercie tous les groupes de heavy metal de la planète pour tout le plaisir qu'ils me procurent à travers leurs disques et leurs concerts :

"Heavy Metal is the law that keeps us all united free
A law that shatters earth and hell
Heavy Metal can't be beaten by any dynasty
We're all wizards fightin' with our spell"
(M. Weikath, K. Hansen)

Table des matières

Introduction générale

Cadre et sujet de cette thèse

Une équation différentielle linéaire homogène d'ordre n (n entier naturel) peut s'écrire sous la forme

$$a_n\, y^{(n)} + a_{n-1}\, y^{(n-1)} + \cdots + a_1\, y' + a_0\, y \;=\; 0, \tag{1}$$

où les a_i sont des fonctions de la variable x et où $y^{(i)}$ désigne la dérivée i-ème de y par rapport à x. De telles équations apparaissent naturellement lorsque l'on cherche à décrire certains phénomènes physiques et ont, par conséquent, été largement étudiées. Si l'on s'intéresse au problème suivant : étant donné les coefficients $a_0, \ldots, a_n$, déterminer les « solutions » de l'équation (1), alors le terme « solution » doit être préalablement précisé. En effet, selon la forme des solutions que l'on recherche, les outils à utiliser peuvent être très différents. D'autre part, en fonction des propriétés des phénomènes décrits par cette équation que l'on cherche à dégager, certaines formes de solutions seront plus adaptées que d'autres. Dans ce mémoire, nous considèrerons principalement des solutions « sous forme close » c'est-à-dire exprimables de manière exacte à l'aide de fonctions usuelles.

Les méthodes modulaires ont déjà montré leur intérêt dans diverses branches du calcul formel en donnant naissance à de nombreux algorithmes performants (voir [GG99, Chapitre 5]). Par exemple, à ce jour, l'algorithme le plus efficace pour factoriser des polynômes à coefficients dans $\mathbb{Z}$ est l'algorithme modulaire développé par van Hoeij dans [Ho02]. Le schéma de fonctionnement général de ces méthodes modulaires est le suivant :

- Réduire le problème à son analogue en caractéristique p,
- Résoudre le problème en caractéristique p,
- Reconstruire la solution du problème initial en caractéristique zéro.

Un des intérêts majeurs de ces méthodes est que résoudre un problème en caractéristique p est souvent plus simple qu'en caractéristique zéro : le principal avantage est qu'en caractéristique p, la taille des entiers qui interviennent dans les calculs intermédiaires est contrôlée.

La théorie des équations différentielles en caractéristique p a beaucoup été étudiée dans les années soixante-dix et quatre-vingt par des mathématiciens comme Katz, Honda ou les Chudnovsky (voir par exemple [Ka70, Ho81, CC85]). Un thème central de ces travaux était la conjecture de Grothendieck (voir Conjecture 1 dans le chapitre 1 et les références données à cet endroit) sur les p-courbures. Cependant, la littérature

concernant l'algorithmique modulaire des équations différentielles linéaires est assez restreinte : ce n'est que dans les années quatre-vingt-dix, dans les travaux de van der Put ([Pu95, Pu96, Pu97]-voir aussi [PS03, Chapitre 13]) que l'on trouve les premières idées dans cette direction.

Dans cette thèse, nous nous proposons d'étudier et développer une approche algorithmique modulaire pour l'étude des équations différentielles linéaires. Par conséquent, les questions générales se trouvant à la base de ce travail sont du type : peut-on contruire des algorithmes modulaires efficaces pour étudier les équations différentielles linéaires ? Quel type d'informations peut-on espérer d'un calcul modulaire ? Peut-on à l'aide de calculs modulaires améliorer les algorithmes existant ?

Factorisation d'opérateurs différentiels linéaires

Un premier problème naturel à regarder d'un point de vue modulaire est celui de la factorisation d'opérateurs différentiels linéaires. À l'équation différentielle linéaire homogène (1), on associe l'opérateur différentiel linéaire

$$L = a_n\,\partial^n + a_{n-1}\,\partial^{n-1} + \cdots + a_1\,\partial + a_0, \tag{2}$$

agissant sur y. Si les coefficients a_i appartiennent à $K = C(x)$ pour un certain corps de nombre $C \subseteq \overline{\mathbb{Q}}$ (où $\overline{\mathbb{Q}}$ représente la clôture algébrique du corps $\mathbb{Q}$ des nombres rationnels), alors L appartient à l'anneau non-commutatif $K[\partial]$ dans lequel la multiplication est définie par $\partial\, x = x\,\partial + 1$. Cet anneau est euclidien à droite (et à gauche) et factoriser l'opérateur L donne des informations sur les solutions de l'équation (1). Par exemple un facteur à droite d'ordre un $\partial - r$ de L avec r appartenant à $\overline{\mathbb{Q}}(x)$ correspond à une solution exponentielle $y = \exp(\int r)$ de (1), *i. e.*, une solution dont la dérivée logarithmique $y'/y = r$ appartient à $\overline{\mathbb{Q}}(x)$.

Le problème de la réductibilité d'équations différentielles linéaires est apparu dans la deuxième partie du dix-neuvième siècle lorsque l'on a cherché à développer une théorie pour étudier les équations différentielles analogue à celle de Lagrange et Galois pour les équations algébriques. Le problème plus spécifique de la factorisation effective d'opérateurs différentiels linéaires a été très étudié depuis le début des années quatre-vingt-dix. À ce jour, il existe essentiellement trois types de méthodes pour aborder ce problème. La première remonte aux travaux de Beke à la fin du dix-neuvième siècle ([Be1894]) et a été améliorée dans les années quatre-vingt-dix ([Sc89, Br94, Ts94]-voir aussi [PS03, 4.2.1]). Elle revient à considérer des puissances extérieures de l'opérateur, à en calculer les solutions exponentielles puis à vérifier des conditions de Plücker. Une deuxième approche est apparue aussi à la fin des années quatre-vingt-dix : van Hoeij ([Ho97a, Ho97b]) a proposé un algorithme calculant tout d'abord des factorisations locales de l'opérateur puis les relevant en des factorisations globales. Une stratégie analogue traitant directement les systèmes différentiels a été développée par Barkatou et Pflügel dans [BP98]. Une troisième méthode communément attribuée à Singer ([Si96]) date elle aussi des années quatre-vingt-dix (voir [Gi98, Ba01, BP98] ou [PS03, 4.2.2]). Elle est basée sur un ensemble, appelé eigenring, que l'on associe à un opérateur différentiel. Cet ensemble peut être vu comme l'anneau des endomorphismes

différentiels associé à l'opérateur et la connaissance d'éléments non-triviaux dans cet ensemble conduit directement à des factorisations de l'opérateur. Notons que les idées à la base de cette troisième approche apparaissent dès les années trente dans les travaux de Jacobson ([Ja37]).

Chapitre 1 : factorisation en caractéristique p

Dans l'optique de construire un algorithme modulaire de factorisation d'opérateurs différentiels, nous nous sommes tout d'abord concentrés sur le problème de la factorisation en caractéristique p qui constitue le sujet du premier chapitre. Nous développons un algorithme de factorisation de systèmes différentiels en caractéristique p. Notons qu'à toute équation différentielle linéaire homogène est associé un système différentiel linéaire (sous forme compagnon) et réciproquement, à tout système différentiel linéaire est associée (via un vecteur cyclique) une équation différentielle linéaire homogène (voir [CK02]).

Pour ce faire, nous nous sommes tout d'abord intéressés aux travaux de van der Put ([Pu95, Pu96, Pu97]), travaux qui à notre connaissance étaient alors les seuls existants sur le problème de la factorisation d'opérateurs différentiels en caractéristique p. Dans [Pu95], l'auteur montre qu'en caractéristique p, les modules différentiels sont entièrement classifiés par la forme de Jordan d'un objet qui leur est associé : la p-courbure. À partir de cette classification, le même auteur a montré dans [Pu97] comment factoriser des opérateurs différentiels linéaires scalaires en caractéristique p. Ces travaux ont résolu théoriquement le problème de la factorisation d'opérateurs différentiels en caractéristique p. Cependant, un travail algorithmique restait nécessaire, notamment pour traiter directement les systèmes : ceci constitue le thème central de cette partie.

Pour développer un algorithme de factorisation de systèmes différentiels linéaires en caractéristique p, nous utilisons les idées de [Pu97] mais nous exhibons aussi les liens entre la p-courbure et l'eigenring ce qui nous permet de suivre l'approche proposée par Barkatou dans [Ba01] qui ne dépend pas de la caractéristique. Nous comparons différentes stratégies, fournissons une implantation (voir Annexe) et une analyse de complexité. Nous contribuons à l'amélioration de la méthode de van der Put : une fois que la décomposition isotypique de l'opérateur a été calculée alors, contrairement à l'algorithme de van der Put, nous utilisons l'eigenring tout entier pour éviter des calculs potentiellement coûteux dans des extensions algébriques. Ceci mène à des méthodes rationnelles pour calculer à la fois une décomposition maximale et une réduction maximale des blocs indécomposables.

Parallèlement à notre travail, Giesbrecht et Zhang ont donné dans [GZ03] un algorithme de factorisation de polynômes de Ore (une classe d'opérateurs contenant les opérateurs différentiels linéaires) en caractéristique p. Toute cette partie constitue une version plus détaillée de l'article [Cl03].

Factorisation de systèmes d'équations aux dérivées partielles D-finis en caractéristique p

Toujours dans ce premier chapitre, nous généralisons les méthodes décrites ci-dessus

pour développer un algorithme de factorisation de systèmes d'équations aux dérivées partielles D-finis en caratéristique p. Le problème de la factorisation effective de tels systèmes a été étudié récemment par Li, Schwarz et Tsarëv qui ont montré comment adapter l'algorithme de Beke dans ce cas (voir [LST02, LST03] et aussi [Ts98, Ts01]). De tels systèmes peuvent s'écrire sous la forme $S = \{\partial_i(Y) = A_i\, Y\}_{1\leq i\leq m}$ où les A_i sont des matrices carrées à coefficients dans $\overline{\mathbb{F}}_p(x_1, \ldots, x_m)$ et $\partial_i := \frac{d}{dx_i}$. La D-finitude implique que les systèmes considérés vérifient les conditions d'intégrabilité données par : $\partial_i(A_j) - \partial_j(A_i) + A_i\, A_j - A_j\, A_i = 0$. Nous montrons comment l'algorithme que nous avons développé dans le cas différentiel ordinaire, c'est-à-dire dans le cas où nous n'avons qu'une seule dérivation, peut s'appliquer ici pour factoriser individuellement chaque $\partial_i(Y) = A_i\, Y$. Ensuite nous utilisons l'hypothèse de D-finitude pour ramener le problème à celui de la réduction simultanée de matrices qui commutent. Nous apportons aussi notre contribution à l'étude de ce problème en proposant une nouvelle approche consistant à considérer une combinaison linéaire à coefficients indéterminés des matrices que nous cherchons à réduire simultanément.

Interlude : reconstruire des factorisations en caractéristique zéro

La dernière étape d'une approche modulaire pour factoriser les opérateurs différentiels consiste à reconstruire, à partir de factorisations modulo p, des factorisations en caractéristique zéro. Une fois encore, les seuls travaux existants sur ce sujet sont ceux de van der Put. Dans [Pu96], il propose un algorithme modulaire pour factoriser les opérateurs différentiels d'ordre 2. Cependant comme l'auteur le fait remarquer cet algorithme possède certaines limitations (à la fois théoriques et algorithmiques) qui font que tel quel il ne peut être complètement satisfaisant. Dans [Pu97], le même auteur propose d'utiliser l'algorithme L.L.L. (voir [GG99, Chapitre 16] ou [LLL82]) pour reconstruire les facteurs mais ici encore, ceci ne semble pas mener à un algorithme modulaire complet efficace.

Si l'on examine le problème analogue dans le monde commutatif, *i. e.*, la factorisation de polynômes alors on remarque que les algorithmes modulaires (Berlekamp-Zassenhaus [GG99, Chapitres 14 et 15]-[LLL82, Ho02]) suivent principalement le schéma suivant :

- réduire les coefficients du polynôme modulo un nombre premier p,
- calculer les facteurs du polynôme modulo p (algèbre linéaire),
- déduire des facteurs p-adiques (lemme de Hensel, [GG99, 15.4]),
- recombiner les facteurs.

La terminaison des algorithmes de ce type repose sur l'existence de la borne de Mignotte (voir [GG99, Corollary 6.33]) sur la taille des coefficients des facteurs.

Si l'on cherche à adapter ce schéma à notre cas, alors à partir de la troisième étape, plusieurs problèmes majeurs apparaissent. Tout d'abord, une remontée type Hensel dans le cas des opérateurs différentiels semble être beaucoup moins simple que dans le cas des polynômes. En effet, supposons qu'un opérateur différentiel L admette pour développement p-adique $L = L_0 + p\, L_1 + \cdots$ avec les $L_i \in \mathbb{F}_p(x)[\partial]$ et supposons que l'on dispose d'une factorisation mod p, $L_0 = M_0\, N_0$. Si nous voulons relever cette factorisation en une factorisation modulo p^2 alors nous sommes ramenés à trouver M_1

et N_1 dans $\mathbb{F}_p(x)[\partial]$ tels que $L_0 + p\,L_1 = (M_0 + p\,M_1)\,(N_0 + p\,N_1) \mod p^2$: il vient donc $M_1\,N_0 + M_0\,N_1 = L_1$ de sorte que M_1 et N_1 sont solutions d'une équation différentielle mixte (voir [Ho96]). On voit ainsi qu'à chaque étape d'une remontée type Hensel on est ramené à résoudre une équation différentielle mixte ce qui s'avère assez coûteux. Nous ne disposons donc pas actuellement d'un procédé efficace nous permettant de réaliser une remontée type Hensel. Notons que pour ce problème, une solution pourrait être de considérer plutôt que des équations différentielles modulo p, des équations différentielles itératives : cette notion introduite par Hasse et Schmidt dans les années trente a été reprise récemment par Matzat et van der Put pour construire une (bonne) théorie de Picard-Vessiot en caractéristique positive ([Ma01, MP02] ou [PS03, 13.3.2]). D'autre part, la phase de recombinaison est elle aussi beaucoup plus compliquée, essentiellement à cause du fait que, contrairement au cas des polynômes, la factorisation d'un opérateur différentiel n'est pas unique. Par exemple $\partial^2 = (\partial + \frac{1}{x+c})\,(\partial - \frac{1}{x-c})$ pour toute constante c dans $\mathbb{Q}$. Cette non-unicité de la factorisation rend tout aussi difficile la construction d'une borne type Mignotte dans le cas différentiel.
En conséquence de ces problèmes, nous allons par la suite chercher à dégager le type d'informations pertinentes que l'on peut espérer obtenir d'un calcul modulaire et, utiliser ces informations pour améliorer certains algorithmes existants.

Chapitre 2 : solutions exponentielles d'équations différentielles linéaires

Nous nous intéressons ici au problème du calcul de solutions exponentielles ou de manière équivalente de facteurs à droite d'ordre 1. Nous considérons donc une équation différentielle homogène de la forme (1) ayant des coefficients dans $C(x)$ où C est un corps de nombre quelconque. Pour calculer les solutions exponentielles d'une telle équation, l'algorithme communément utilisé est celui de Beke qui remonte à la fin du dix-neuvième siècle ([Be1894]-voir aussi [Ho97b, 3.4] ou [PS03, 4.1]). Un algorithme calculant directement les solutions exponentielles d'un système différentiel linéaire a été développé par Pflügel dans [Pf97].

L'algorithme de Beke peut être décrit comme suit : tout d'abord, en chaque singularité de l'équation différentielle, nous calculons des objets appelés exposants généralisés qui ont déjà été utilisés par van Hoeij dans [Ho97a, Ho97b]. Cette notion d'exposants généralisés est très proche de la notion, plus connue, de « parties exponentielles locales ». Notons qu'en chaque singularité, ces exposants généralisés sont en nombre fini égal à n. Ensuite pour chaque combinaison d'exposants généralisés (choisir un exposant généralisé à chaque singularité), le problème est ramené au calcul des solutions polynomiales (voir Chapitre 4) d'une équation différentielle auxiliaire. Tel quel, cet algorithme possède deux principaux inconvénients :

- un problème combinatoire : si l'on note m le nombre de singularités, nous avons alors au plus n^m combinaisons d'exposants généralisés à tester,
- un problème d'extensions algébriques : les singularités et les exposants généralisés sont définis sur des extensions algébriques du corps de nombre C et par conséquent

les combinaisons peuvent être définies sur de grandes extensions algébriques de C : le degré de ces extensions peut atteindre $m!\,n^m$ (si on procède naivement). Considérons l'opérateur différentiel suivant :

$$L = 9(x^3-2)^5\partial^3 + (x^3-2)(2x^{10} - 12x^7 + 108x^5 + 24x^4 - 216x^2 - 16x - 9)\partial \\ -2x(190x^6 - 274x^3 - 27x - 212).$$

L possède trois singularités à distance finie à savoir les racines de x^3-2. En chacune de ces 3 singularités les exposants généralisés sont 2, $\frac{1}{36}\frac{\alpha^2}{(x-\alpha)} - \frac{1}{18}\alpha$ et $-\frac{1}{36}\frac{\alpha^2}{(x-\alpha)} + 4 + \frac{1}{18}\alpha$ où $\alpha := \mathrm{RootOf}(x^3-2)$. À l'infini, les exposants généralisés sont $0, -\frac{5}{3}, -\frac{4}{3}$. Par conséquent, l'algorithme de Beke aura $3^4 = 81$ combinaisons à tester et certaines d'entre elles sont définies sur des extensions algébriques de degré 6 de $\mathbb{Q}$.

Dans ce deuxième chapitre, qui constitue un article écrit avec M. van Hoeij (voir [CH04]), nous présentons un nouvel algorithme pour calculer les solutions exponentielles d'équations différentielles linéaires ayant des coefficients dans $C(x)$ où C est un corps de nombre quelconque. L'idée à la base de notre algorithme est de combiner des informations locales et modulaires. Plus précisément, nous prouvons les liens entre la réduction modulo p des exposants généralisés et la p-courbure ce qui nous permet d'utiliser des calculs modulaires pour réduire le nombre de combinaisons à tester dans l'algorithme de Beke. Ceci s'avère d'autant plus intéressant si les combinaisons éliminées sont celles définies sur de grandes extensions algébriques. Pour pouvoir utiliser ces calculs modulaires, nous avons besoin d'éviter certains nombres premiers : nous montrons donc comment calculer un « bon » nombre premier.

Revenons à l'opérateur précédent. Le nombre premier $p = 5$ est un bon nombre premier pour L. En cherchant parmi les exposants généralisés ceux qui coïncident avec l'information encodée par la p-courbure, nous pouvons conclure qu'une solution exponentielle de L ne peut venir que d'une combinaison où nous avons choisi l'exposant généralisé $\frac{1}{36}\frac{\alpha^2}{(x-\alpha)} - \frac{1}{18}\alpha$ en la singularité α. Ceci réduit donc le nombre de possibilités à tester de 81 à 3 et de plus toutes les combinaisons définies sur des extensions algébriques de $\mathbb{Q}$ ont été éliminées (par un argument de trace).

Ensuite nous proposons une méthode pour aborder le problème du corps (ici aussi les informations modulaires peuvent s'avérer utiles) : nous montrons comment trouver les extensions de C sur lesquelles des solutions exponentielles peuvent être définies. À partir de cela, nous donnons un algorithme complet calculant une base de solutions exponentielles d'une équation différentielle linéaire.

Chapitre 3 : solutions hypergéométriques d'équations aux différences linéaires

Une fois le travail sur les solutions exponentielles achevé, la question naturelle que nous nous sommes posés a été la suivante : peut-on utiliser le même genre de techniques c'est-à-dire combiner des informations locales et modulaires pour améliorer l'algorithme de calcul de solutions hypergéométriques d'équations aux différences linéaires ?

Une équation aux différences linéaire d'ordre n peut s'écrire sous la forme

$$L(u(x)) = a_n(x)\,u(x+n) + a_{n-1}(x)\,u(x+n-1) + \cdots + a_0(x)\,u(x) = 0, \qquad (3)$$

où les a_i sont des éléments de $C(x)$ pour un certain corps de nombre C. Les solutions hypergéométriques sont l'analogue des solutions exponentielles à savoir les fonctions u satisfaisants (3) et telles que $u(x+1)/u(x) \in \overline{\mathbb{Q}}(x)$. Depuis le début des années quatre-vingt-dix, l'algorithme utilisé pour calculer les solutions hypergéométriques est celui développé par Petkovšek dans [Pe92] (voir aussi [PWZ96]). Bien qu'ayant d'intéressantes applications, en pratique, cet algorithme est limité car il calcule dans le corps de décomposition de $a_0\,a_n$ qui est souvent très gros. De plus, même lorsque le corps de décomposition n'est pas trop gros, l'algorithme n'est pas optimal puisqu'il considère plus de combinaisons qu'il n'est nécessaire : pour être plus précis, il prend en compte toutes les combinaisons comprenant un facteur (pas nécessairement irréductible) de a_0 et un facteur de a_n.

Nous reprenons ici l'exemple donné par van Hoeij dans [Ho99, 1.5]. Considérons l'équation aux différences linéaires

$$a_3(x)\,u(x+3) + a_2(x)\,u(x+2) + a_1(x)\,u(x+1) + a_0(x)\,u(x) = 0,$$

où :
$a_0(x) = 18\,(2\,x+3)\,(x+2)\,(140\,x^3 + 1151\,x^2 + 3114\,x + 2781)\,(x+1)^2$,
$a_1(x) = -(x+2)\,(23660\,x^6 + 302879\,x^5 + 1581604\,x^4 + 4314577\,x^3 + 6487290\,x^2 + 5099454\,x + 1638144)$,
$a_2(x) = (18380306\,x^2 + 13291032\,x + 237304\,x^6 + 1637876\,x^5 + 4046652 + 14560\,x^7 + 6200310\,x^4 + 13887720\,x^3)$,
$a_3(x) = -4(140\,x^3 + 731\,x^2 + 1232\,x + 678)\,(2\,x+7)^2\,(x+3)^2$.

Pour cet opérateur, l'algorithme de Petkovšek est amené à considérer 6912 combinaisons et à calculer dans le corps de décomposition de $a_0\,a_3$. Ceci entraine de nombreux calculs coûteux qui rendent l'algorithme peu performant sur cet exemple.

Pour parer à ces problèmes, van Hoeij a introduit dans [Ho99] le concept de singularité à distance finie : ces singularités sont les racines de $a_0\,a_n$ modulo $\mathbb{Z}$. En chaque singularité à distance finie, il a défini, par analogie avec la notion d'exposants généralisés, des types locaux. En prenant aussi en compte une notion de type local à l'infini, il a exhibé des relations de Fuchs pour les équations aux différences (notons qu'il a ici deux relations de Fuchs contre une seule dans le cas différentiel). La conséquence de son travail est une adaptation de l'algorithme de Beke, décrit au chapitre 2, aux équations aux différences linéaires.

Revenons à l'exemple précédent. Pour cet exemple les racines de a_3 sont $-7/2$, -3, α_1, α_2, α_3 où les α_i sont les racines de $140\,x^3 + 731\,x^2 + 1232\,x + 678$ et celles de a_0 sont $-3/2$, -2, -1, $\alpha_1 - 1$, $\alpha_2 - 1$, $\alpha_3 - 1$. Les singularités à distance finies sont donc $p_i = \alpha_i + \mathbb{Z}$ pour $i \in \{1,2,3\}$, $p_4 = 0 + \mathbb{Z}$ et $p_5 = 1/2 + \mathbb{Z}$. En chacune d'elles, nous calculons alors les types locaux : ici nous trouvons $\{0\}$ en p_1, p_2, p_3, $\{-2,-1,0,1,2\}$ en p_4 et $\{-2,-1,0,1\}$ en p_5. Par conséquent il n'y aura que 20 combinaisons à tester dans un algorithme type Beke. Notons encore que les 20 combinaisons en question ne feront

intervenir aucune extension algébrique. Dans cette explication nous n'avons pas pris en compte les types locaux à l'infini ce qui grâce aux relations de Fuchs qui en découlent aurait encore diminuer le nombre de cas à tester.

Ce chapitre 3 fait aussi l'objet d'un article (soumis) écrit avec M. van Hoeij. Les contributions que nous apportons au problème sont de natures diverses. Tout d'abord, nous donnons des détails précis sur comment implanter de manière efficace l'algorithme de [Ho99]. Notons qu'une version très proche de l'algorithme donné dans ce chapitre a été implantée par van Hoeij et est disponible dans Maple 9. Une méthode pour éviter les calculs dans le corps de décomposition de $a_0\,a_n$ avait été proposée dans [Ho99] lorsque l'ordre de l'opérateur est inférieur ou égal à 3. Nous généralisons ici ces résultats à un ordre quelconque. Ensuite, nous montrons comment obtenir les mêmes filtres modulaires que dans le cas du calcul des solutions exponentielles. Pour ce faire, nous prouvons les liens entre la réduction modulo p des types locaux et la p-courbure de l'équation aux différences réduite modulo p. Nous décrivons les améliorations de l'algorithme qui en découlent. Nous terminons ce chapitre en montrant comment tout ce qui a été fait ici pourrait être fait de la même manière dans le cas des équations aux q-différences. Pour ceci nous introduisons les notions nouvelles de types locaux et relations de Fuchs pour les équations aux q-différences. La conséquence immédiate de ces nouvelles définitions est un algorithme analogue à celui développé dans ce chapitre pour calculer les solutions q-hypergéométriques d'équations aux q-différences linéaires.

Chapitre 4 : solutions polynomiales d'équations différentielles linéaires

Le quatrième chapitre de cette thèse est consacré au calcul de solutions polynomiales d'équations différentielles linéaires. Ce problème, déjà étudié par Liouville au dix-neuvième siècle, est particulièrement important car il se trouve être une sous-procédure indispensable à de nombreux algorithmes en calculs formels. Par exemple, comme nous l'avons vu au chapitre 2, calculer les solutions exponentielles d'une équation différentielle linéaire se ramène au calcul des solutions polynomiales d'équations différentielles linéaires auxiliaires. Actuellement, les deux principaux algorithmes utilisés sont celui de Abramov, Bronstein et Petkovšek ([ABP95]) et celui de Barkatou ([Ba97, Ba99]). Ces deux algorithmes fonctionnent tous les deux de la manière suivante : tout d'abord on calcule les degrés possibles pour d'éventuelles solutions polynomiales puis on calcule les coefficients des solutions cherchées. Dans cette partie, nous détaillons ces deux algorithmes, étudions leur complexité et les comparons.

La principale contribution de ce chapitre est le développement d'un algorithme modulaire pour calculer les solutions polynomiales d'une équation différentielle linéaire. Contrairement aux algorithmes des chapitres deux et trois qui utilisent l'information modulaire comme filtre, celui-ci est entièrement modulaire. Nous montrons tout d'abord que les deux algorithmes cités précédemment peuvent être adapté à la caractéristique p. Nous pouvons ainsi, à l'aide de la version modulaire de l'un de ces algorithmes, calculer les solutions polynomiales de l'équation réduite modulo certains nombres premiers. Nous reconstruisons ensuite les solutions de l'équation en caractéristique zéro

en utilisant une version rationnelle de l'algorithme des restes chinois ([GG99, 5.10]) qui fonctionne comme suit : tout d'abord avec l'algorithme (classique) des restes chinois ([GG99, 5.4]), nous obtenons des polynômes à coefficients entiers puis, à partir de ces polynômes, nous reconstruisons des polynômes à coefficients rationnels à l'aide de la méthode de reconstruction rationnelle ([GG99, 5.10] ou [CE95]). Pour prouver la correction de ce procédé, nous exhibons une borne *a priori* sur la taille des coefficients des solutions polynomiales d'une équation différentielle linéaire. Une alternative à cette méthode consiste à calculer les solutions polynomiales modulo un seul premier p et à les relever à l'aide d'un relèvement type Hensel. Notons qu'à notre connaissance cet algorithme est le premier algorithme entièrement modulaire dédié aux équations différentielles. La complexité de cet algorithme modulaire est étudiée, l'algorithme est implanté et nous comparons différentes stratégies.

Notre approche modulaire fournit d'intéressants résultats théoriques qui mènent à des tests de non-existence de solution polynomiale non-nulle très utiles en pratique.

Considérons par exemple l'opérateur différentiel linéaire suivant :

$$L = (x^2+x+8)^2\,\partial^4 + (x^2+x+8)\,(2\,x^8 - 5814\,x - 5331)\,\partial^3 + (x^{16} + 256\,x^7 + 23261828\,x + 10595524 - 8696\,x^9 + 12684698\,x^2 - 7963\,x^8)\,\partial^2 + (-25039\,x^{12} + 359649\,x^{10} - 1163014557 + 11410642\,x^5 - 621906702\,x^3 - 159337\,x^{11} + 23047\,x^{13} + 6957787\,x^7 + 8819879\,x^8 + 249\,x^{14} + 530621566\,x^4 - 2888\,x^{15} - 67752682\,x^6 - 3710072636\,x - 3623065826\,x^2 + 915047\,x^9)\partial + 2086568\,x^{14} - 35187260057\,x^5 - 359808\,x^{13} - 2959136216\,x^8 + 288366200961\,x^2 - 828786528\,x^9 + 5642839360\,x^7 + 12468795209\,x^6 + 47582358547 + 199846009415\,x - 46102704943\,x^4 + 39075760\,x^{11} + 130179393143\,x^3 + 360154648\,x^{10} - 33298936\,x^{12}.$$

Le degré d'éventuelles solutions polynomiales appartient à l'ensemble $\{1441, 1448\}$. Par conséquent, les algorithmes usuels mettent beaucoup de temps pour arriver à la conclusion que cet opérateur ne possède pas de solution polynomiale non-nulle. Avec notre test modulaire, nous sommes en mesure de détecter cette non-existence à faible coût : un calcul modulo $p = 3$ est suffisant.

Ces tests modulaires de non-existence de solution polynomiale non-nulle sont très utiles en pratique : en effet, dans la plupart des problèmes se ramenant au calcul des solutions polynomiales d'opérateurs auxiliaires, ces opérateurs auxiliaires possèdent des coefficients beaucoup plus gros que ceux de l'opérateur de départ et passent les tests de non-existence contenus dans les algorithmes classiques (car ils possèdent des exposants entiers positifs en chaque singularité). Par exemple l'opérateur L donné au-dessus est l'un des opérateurs dont nous sommes amené à chercher les solutions polynomiales lorsque l'on cherche à calculer toutes les solutions exponentielles de l'opérateur

$$(x^2+x+8)^2\,\partial^4 - (x^2+x+8)\,(2\,x^8 - 6\,x - 13)\,\partial^3 + (-28\,x^8 + 76 + 29\,x + x^{16} - 128\,x^7 + 8\,x^2 - 20\,x^9)\,\partial^2 + (-448\,x^6 - 66\,x^8 + 2\,x - 104\,x^7 + 8\,x^{15} + 11)\,\partial.$$

Une partie de ce chapitre constitue un travail réalisé en collaboration avec A. Fredet.

Chapitre 1

Factorisation de systèmes différentiels en caractéristique p

Introduction

Le problème de la factorisation d'opérateurs différentiels scalaires $L = \sum_{i=0}^{n} a_i \, (d/dx)^i$ avec les $a_i \in \mathbb{Q}(x)$, ou plus généralement de systèmes différentiels linéaires homogènes $Y' = A\,Y$ où A est une matrice carrée de taille n à coefficients dans $\mathbb{Q}(x)$, est un problème important et non-trivial en calcul formel. De nos jours, il existe essentiellement trois types de méthodes pour aborder ce problème. La première remonte à Beke (voir [Be1894]) et revient à considérer des puissances extérieures du système, à en calculer les solutions exponentielles puis à vérifier les relations de Plücker. Des améliorations algorithmiques de cette méthode sont données dans [Sc89, Br94, Ts94] (voir aussi [PS03, Section 4.2.1] ou encore [CW04, Appendix A]). Une deuxième approche ([Ho97a, Ho97b] pour le cas scalaire, [BP98] pour les systèmes) étudie les factorisations locales et les relève en des factorisations globales. Une troisième méthode ([Si96, Ba01, BP98] ou [PS03, Section 4.2.2]) dont les idées de base apparaissaient déjà dans [Ja37] est basée sur le calcul et l'utilisation de l'anneau des endomorphismes de l'espace des solutions du système, appelé *eigenring*. L'existence de diviseurs de zéro (resp. d'idempotents) dans cet anneau permet de réduire (resp. décomposer) le système différentiel.

Le fait que les meilleurs algorithmes de factorisation de polynômes soient basés sur des méthodes modulaires (Berlekamp-Zassenhaus [GG99, Section 14.8], [LLL82] ou encore [Ho02]), nous amène à penser qu'une approche similaire pourrait être satisfaisante dans notre cas. De plus, en pratique, les algorithmes existants sont limités par les dimensions (du système et de ses coefficients) du problème et, excepté dans [Gr90], nous n'avons pas d'analyse de complexité. En fait, comme pour la factorisation de polynômes (voir le chapitre 4), il est intéressant d'avoir plusieurs algorithmes car la diversité des situations entraine qu'aucun algorithme n'est universellement meilleur que les autres. Toutes ces raisons constituent les principales motivations pour développer des méthodes modulaires dans le cas différentiel. Ce chapitre est un premier pas dans cette direction : on s'intéresse à la factorisation de systèmes différentiels linéaires à coefficients dans

$\mathbb{F}_p(x)$.

La théorie des équations différentielles en caractéristique p prend ses racines dans des travaux comme [Ka70, Ho81, Ka82, CC85]. Plus récemment, van der Put a donné dans [Pu95] (voir aussi [PS03, 13.1]) une classification des modules différentiels en caractéristique p. Ceci fournit naturellement un algorithme complet (voir [Pu97]) pour factoriser les opérateurs différentiels en caractéristique p : cet algorithme est basé sur le fait que la forme de Jordan de la *p-courbure* donne toutes les factorisations de l'opérateur. Dans [GZ03], Giesbrecht et Zhang présentent une méthode alternative pour factoriser et décomposer des polynômes de Ore (une classe d'objets contenant les opérateurs différentiels) à coefficients dans $\mathbb{F}_p(x)$. Leurs algorithmes reposent sur les diviseurs de zéro ainsi que les idempotents contenus dans l'eigenring ; ils montrent comment calculer de tels éléments en utilisant l'algorithme développé dans [IRS94] et comment en déduire des factorisations du polynôme de Ore.

La première contribution de ce chapitre est une comparaison de différentes stratégies, une analyse de complexité ainsi qu'une implantation (Voir Appendice). Pour développer un algorithme de factorisation de systèmes différentiels en caractéristique p, nous nous basons sur la classification des modules différentiels en caractéristique p ce qui nous amène à réduire la p-courbure du système. Certains résultats pouvant se voir à partir de cette classification sont re-démontrés d'une manière élémentaire et algorithmique ce qui nous permet d'implanter un algorithme de factorisation de systèmes et de faciliter la généralisation de notre algorithme à d'autres cas et notamment au cas des systèmes d'équations aux dérivées partielles. Nous établissons aussi les liens entre la p-courbure et l'eigenring du système ; nous pouvons ainsi présenter notre algorithme en suivant la méthode développée par Barkatou dans [Ba01] qui ne dépend pas de la caractéristique. Nous contribuons aussi à l'amélioration de la méthode de van der Put : une fois que la décomposition isotypique de l'opérateur a été calculée alors, contrairement à l'algorithme de van der Put, nous utilisons l'eigenring tout entier pour nous éviter de calculer dans des extensions algébriques de $\mathbb{F}_p(x)$. Cela mène à des méthodes rationnelles pour calculer à la fois une décomposition maximale et une réduction maximale des blocs indécomposables.

D'une manière tout à fait naturelle, nous généralisons notre algorithme pour obtenir un algorithme donnant les « factorisations locales » de systèmes différentiels en caractéristique p, ainsi qu'un algorithme de factorisation des systèmes d'équations aux différences linéaires en caractéristique p.

Nous nous intéressons ensuite au cas des systèmes d'équations aux dérivées partielles D-finis. Ici, la généralisation de notre algorithme s'avère un peu plus compliquée car on ne dispose plus de la classification des modules différentiels en caractéristique p. Cependant, en tirant parti du fait que nous avons ré-écrits d'une manière élémentaire les preuves de certains résultats pouvant se voir à partir de cette classification, nous montrons comment les généraliser. En cherchant à développer un algorithme de factorisation

analogue à celui donné dans le cas différentiel ordinaire, nous sommes amenés à réduire simultanément une famille de matrices qui commutent ; nous étudions donc ce dernier problème et nous proposons une nouvelle approche en considérant une combinaison linéaire à coefficients indéterminés des matrices à réduire simultanément.

1.1 Généralités

Nous rappelons ici certaines notions utilisées dans la suite de ce chapitre : mis à part la proposition 1, tout ceci est standard (voir [PS03, Ja37, Or33, In26, Si96, Gr90, Ba01]). La plupart des définitions données ne dépendent pas de la caractéristique et sont énoncées pour un corps $\mathcal{F}$ arbitraire.

1.1.1 Corps et systèmes différentiels

Définition 1. *Une application $\partial : \mathcal{F} \to \mathcal{F}$ est une* dérivation sur $\mathcal{F}$ *si elle satisfait les deux conditions suivantes :*

$$\begin{array}{rcl} \forall\, x,\, y \in \mathcal{F}, \quad \partial(x+y) & = & \partial(x) + \partial(y), \\ \partial(x\,y) & = & x\,\partial(y) + \partial(x)\,y. \end{array}$$

Un corps $\mathcal{F}$ muni d'une dérivation ∂ est appelé corps différentiel *et noté $(\mathcal{F}, \partial)$. L'ensemble*

$$\mathrm{Const}(\mathcal{F}) := \{f \in \mathcal{F} \,/\, \partial(f) = 0\}$$

est un corps appelé corps des constantes *de $(\mathcal{F}, \partial)$.*

Exemple 1. *Le corps $k(x)$ des fractions rationnelles en la variable x et à coefficients dans un corps k, muni de la dérivation usuelle $\frac{d}{dx}$ est un corps différentiel. Si k est de caractéristique zéro, alors le corps des constantes de $(k(x), \frac{d}{dx})$ est k. En revanche si k est de caractéristique $p > 0$, le corps des constantes de $(k(x), \frac{d}{dx})$ est $k(x^p)$.*

Soit $(\mathcal{F}, \partial)$ un corps différentiel. Pour une matrice quelconque $A = (a_{ij})_{i,j}$ avec les $a_{i,j}$ dans $\mathcal{F}$, nous notons $\partial(A)$ la matrice $(\partial(a_{i,j}))_{i,j}$. Dans la suite, nous considérons uniquement des systèmes différentiels linéaires (homogènes) de la forme $\partial(Y) = A\,Y$ avec $A \in \mathbb{M}_n(\mathcal{F})$, l'ensemble des matrices carrées de taille n à coefficients dans $\mathcal{F}$. Un système $\partial(Y) = A\,Y$ sera noté $[A]$. L'entier n est appelé *dimension du système* $[A]$.

Soit $P \in \mathrm{GL}_n(\mathcal{F})$, le sous-ensemble de $\mathbb{M}_n(\mathcal{F})$ formé des matrices inversibles. Si l'on substitue $Y = P\,Z$ dans l'équation $\partial(Y) = A\,Y$, alors Z vérifie le système

$$\partial(Z) \;=\; P^{-1}(A\,P - \partial(P))\,Z,$$

ce qui justifie la définition suivante :

Définition 2. *Soit $(\mathcal{F}, \partial)$ un corps différentiel. Deux systèmes différentiels linéaires $[A]$ et $[B]$ avec $A, B \in \mathbb{M}_n(\mathcal{F})$ sont dits* équivalents sur $\mathcal{F}$ *s'il existe une matrice $P \in \mathrm{GL}_n(\mathcal{F})$ telle que*

$$B = P^{-1}(A\,P - \partial(P)).$$

Dans la suite, pour alléger les notations, nous poserons

$$P[A] := [P^{-1}(A\,P - \partial(P))].$$

1.1.2 Opérateur et module différentiel associé

À un système différentiel $[A]$ avec $A \in \mathbb{M}_n(\mathcal{F})$, nous associons l'opérateur

$$\Delta_A := \partial - A : \mathcal{F}^n \to \mathcal{F}^n.$$

Cet opérateur vérifie la condition de Leibniz : pour tout couple (f, v) dans $\mathcal{F} \times \mathcal{F}^n$, $\Delta_A(f\,v) = \partial(f)\,v + f\,\Delta_A(v)$; c'est donc un opérateur différentiel $\mathrm{Const}(\mathcal{F})$-linéaire.

Remarque 1. *Un calcul direct montre que si $B = P[A]$, alors $\Delta_B = P^{-1}\,\Delta_A\,P$.*

Notons $\mathcal{D} := \mathcal{F}[X]$ l'anneau non-commutatif dans lequel la multiplication est définie comme suit : pour tout a dans $\mathcal{F}$, $X\,a := a\,X + a'$. On peut alors munir $\mathcal{F}^n$ d'une structure de module sur $\mathcal{D}$, structure associée à notre système différentiel $[A]$; plus précisément, $(\mathcal{F}^n, \Delta_A)$ est un module différentiel :

Définition 3. *Un* module différentiel (M, Δ) *est la donnée d'un $\mathcal{F}$-espace vectoriel de dimension finie muni d'une application additive $\Delta : M \to M$ vérifiant la condition de Leibniz $\Delta(f\,v) = \partial(f)\,v + f\,\Delta(v)$ pour tout $f \in \mathcal{F}$ et pour tout $m \in M$.*

Un module différentiel (M, Δ) est aussi un $\mathcal{D}$-module sous l'action

$$\forall f \in \mathcal{D},\ \forall m \in M,\ f.m = f(\Delta)\,m.$$

Nous renvoyons à [PS03] pour tout détail concernant les modules différentiels et à [Pu95] pour tout ce qui concerne la structure de module différentiel en caractéristique p.

1.1.3 Correspondance Système/Opérateur scalaire

Soit $(\mathcal{F}, \partial)$ un corps différentiel quelconque. Dans ce chapitre nous nous intéressons aux systèmes différentiels linéaires $\partial(Y) = A\,Y$, mais aussi aux opérateurs différentiels linéaires scalaires à coefficients dans $(\mathcal{F}, \partial)$ c'est-à-dire aux opérateurs de la forme $L = \sum_{i=0}^{n} a_i\,\partial^i$ avec les a_i dans $\mathcal{F}$ et $a_n \neq 0$. Il existe une correspondance entre ces deux classes d'objets : à tout opérateur différentiel linéaire scalaire $L = \sum_{i=0}^{n} a_i\,\partial^i$ d'ordre n on associe le système différentiel linéaire $\partial(Z) = C\,Z$ où

$$C = \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ -a_0/a_n & -a_1/a_n & \cdots & \cdots & \cdots & -a_{n-1}/a_n \end{pmatrix},\quad Z = \begin{pmatrix} y \\ \partial(y) \\ \vdots \\ \partial^{n-1}(y) \end{pmatrix}.$$

Réciproquement, si le corps différentiel $(\mathcal{F}, \partial)$ possède un élément non-constant (et si $p > n$ dans le cas où $\mathcal{F}$ est de caractéristique p), alors, d'après le lemme du vecteur cyclique, à tout système différentiel linéaire $[A]$ à coefficients dans $(\mathcal{F}, \partial)$, on peut associer un opérateur différentiel linéaire scalaire par le procédé qui suit (voir [Ba93, Section 5]).

Définition 4. *Soit $[A]$ un système différentiel linéaire avec $A \in \mathbb{M}_n(\mathcal{F})$. Soit Λ_0 un vecteur (écrit en ligne) de $\mathcal{F}^n$ et considérons la suite de vecteurs $(\Lambda_i)_i$ définie par $\Lambda_{i+1} = \partial(\Lambda_i) + \Lambda_i A$ pour tout $i \geq 1$. On dit que Λ_0 est* un vecteur cyclique *si le déterminant de la matrice $\begin{pmatrix} \Lambda_0 \\ \vdots \\ \Lambda_{n-1} \end{pmatrix} \in \mathbb{M}_n(\mathcal{F})$ est non-nul.*

Nous renvoyons à [CK02] (et aux références données dans cet article) pour les résultats d'existence de vecteurs cycliques sur différents corps différentiels (voir aussi l'exposition faite dans [Co36]).

Remarque 2. *Lorsque le corps différentiel est $(k(x), \frac{d}{dx})$, alors l'existence d'un vecteur cyclique est assurée si k est de caractéristique zéro. Cependant si k est de caractéristique p, alors l'existence d'un vecteur cyclique pour un système différentiel linéaire $[A]$ est assurée seulement si p est supérieur à la dimension n du système (voir [CK02, Section 6]).*

Soit Λ_0 un vecteur cyclique. Pour toute solution Y de $[A]$, $y := \Lambda_0 Y$ vérifie $L(y) = 0$ où $L = \sum_{i=0}^{n} a_i \partial^i$ avec les a_i donnés par :

$$a_n = \det \begin{pmatrix} \Lambda_0 \\ \vdots \\ \Lambda_{n-1} \end{pmatrix}, a_0 = (-1)^n \det \begin{pmatrix} \Lambda_1 \\ \vdots \\ \Lambda_n \end{pmatrix} \text{ et, si } i \notin \{0, n\}, a_i = (-1)^{n-i} \det \begin{pmatrix} \Lambda_0 \\ \vdots \\ \Lambda_{i-1} \\ \Lambda_{i+1} \\ \vdots \\ \Lambda_n \end{pmatrix}.$$

Le fait que Λ_0 soit cyclique induit une bijection entre l'espace des solutions de L et celui de $[A]$.

On en déduit alors la proposition suivante qui nous sera utile pour certaines analyses de complexité.

Proposition 1. *Soit $[A]$ un système différentiel linéaire de dimension n à coefficients dans $(k(x), \frac{d}{dx})$ où k est soit un corps de caractéristique zéro, soit un corps de caractéristique $p > n$. Soit a une borne sur le degré en x des entrées de A. Calculer un vecteur cyclique Λ_0 ainsi que les a_i définis ci-dessus peut se faire en $\mathcal{O}(n^5)$ opérations dans k.*
Soit Λ_0 un vecteur cyclique et soit λ une borne sur le degré en x des entrées de Λ_0. Alors une borne sur le degré en x des coefficients de l'opérateur scalaire L associé est $n\,\lambda + a\,\frac{n\,(n+1)}{2}$.

Démonstration. Tout d'abord, le résultat de complexité provient de [CK02, Section 4]. Maintenant, si $\deg_x(A) \leq a$ et $\deg_x(\Lambda_0) \leq \lambda$, alors pour tout i, le degré en x des Λ_i définis dans la définition 4 est inférieur ou égal à $\lambda + i\,a$. De plus, d'après les formules pour les a_i données ci-dessus, on a : pour tout i, $\deg_x(a_i) \leq n\,\lambda + a\,(\frac{n\,(n+1)}{2} - i)$; d'où le résultat. □

1.1.4 D'autres définitions

Dans certaines preuves de ce chapitre, nous aurons besoin de considérer l'adjoint du système différentiel linéaire $[A]$.

Définition 5. *Soit $(\mathcal{F}, \partial)$ un corps différentiel et $A \in \mathbb{M}_n(\mathcal{F})$. Le* système adjoint *noté $[A^*]$ du système $[A]$ est le système $[-\,{}^tA]$ où ${}^tA$ désigne la transposée de la matrice A.*

Notons que si Y est une matrice fondamentale de solutions de $[A]$, alors $({}^tY^{-1})$ est une matrice fondamentale de solutions de $[A^*]$. En effet, si $Y' = A\,Y$, alors $0 = (Y\,Y^{-1})' = A + Y\,(Y^{-1})'$ d'où $(Y^{-1})' = -Y^{-1}\,A$ et on conclut en prenant la transposée de cette relation.

Dans ce qui suit nous utiliserons les notions de *solutions rationnelles et algébriques* d'un système différentiel linéaire.

Définition 6. *Soit $(\mathcal{F}, \partial)$ un corps différentiel et $A \in \mathbb{M}_n(\mathcal{F})$.*
On appelle solution rationnelle de $[A]$ *un vecteur $Y \in \mathcal{F}^n$ satisfaisant $Y' = A\,Y$.*
On appelle solution algébrique de $[A]$ *un vecteur $Y \in \mathcal{F}^n$ vérifiant $Y' = A\,Y$ et dont les entrées appartiennent à une extension algébrique de $\mathcal{F}$.*
Lorsque $\mathcal{F} = k(x)$ avec k corps quelconque, alors on appelle solution polynomiale de $[A]$ *un vecteur $Y \in k[x]^n$ satisfaisant $Y' = A\,Y$.*

Remarque 3. *Lorsque $\mathcal{F} = k(x)$ où k est un corps de caractéristique p, l'existence de solutions rationnelles est équivalente à l'existence de solutions polynomiales : en effet, si $Y = {}^t(y_1, \ldots, y_n) \in k(x)^n$ est une solution rationnelle de $[A]$ et si D est le plus petit commun multiple des dénominateurs des y_i élevé à la puissance p, alors $D\,Y \in k[x]^n$ est une solution polynomiale de $[A]$. Notons, pour être complet, que dans ce cas l'existence de solutions rationnelles est aussi équivalente à l'existence de solutions sous forme de séries (voir [Hon81, Lemma 1]),* i. e.*, dans $k((x))^n$.*

1.1.5 Factorisation de systèmes différentiels

Définition 7. *Soit $(\mathcal{F}, \partial)$ un corps différentiel et $A \in \mathbb{M}_n(\mathcal{F})$. Le système différentiel linéaire $[A]$ est dit* réductible sur $\mathcal{F}$ *s'il existe un système équivalent (sur $\mathcal{F}$) $[B]$ avec*

$$B = \begin{pmatrix} B^{[1]} & B^{[3]} \\ 0 & B^{[2]} \end{pmatrix},$$

où $B^{[1]}$ et $B^{[2]}$ sont deux matrices carrées de taille strictement inférieure à n.
Si $[A]$ n'est pas réductible sur $\mathcal{F}$, alors il est dit irréductible sur $\mathcal{F}$.

Dans le cas où $[A]$ est réductible, $[B^{[2]}]$ est appelé *un facteur* de $[A]$. Cette terminologie est justifiée par le fait que si Z est solution de $\partial(Z) = B^{[2]}\,Z$, alors ${}^t(0, Z)$ est solution de $[B] = P[A]$ de sorte que $P\,{}^t(0, Z)$ est solution de $[A]$.

Définition 8. *Soit $(\mathcal{F}, \partial)$ un corps différentiel et $A \in \mathbb{M}_n(\mathcal{F})$. Le système différentiel linéaire $[A]$ est dit* décomposable sur $\mathcal{F}$ *s'il existe un système équivalent (sur $\mathcal{F}$) $[B]$ avec*

$$B = \begin{pmatrix} B^{[1]} & 0 \\ 0 & B^{[2]} \end{pmatrix} := \bigoplus_{i=1}^{2} B^{[i]},$$

où $B^{[1]}$ et $B^{[2]}$ sont deux matrices carrées de taille strictement inférieure à n.
Dans la suite, nous noterons aussi $[B] = \bigoplus_{i=1}^{2}[B^{[i]}]$.
Si $[A]$ n'est pas décomposable sur $\mathcal{F}$, alors il est dit indécomposable sur $\mathcal{F}$.

Dans le cas où $[A]$ est décomposable, $[B^{[1]}]$ et $[B^{[2]}]$ sont tous les deux des *facteurs* de $[A]$.

Définition 9. *Soit $(\mathcal{F}, \partial)$ un corps différentiel et $A \in \mathbb{M}_n(\mathcal{F})$. Le système différentiel linéaire $[A]$ est dit* complètement réductible sur $\mathcal{F}$ *s'il existe un système équivalent (sur $\mathcal{F}$) $[B]$ avec*

$$[B] = \bigoplus_{i=1}^{r} [B^{[i]}],$$

où $r \geq 1$ et où chaque $[B^{[i]}]$ $(i = 1, \ldots, r)$ est irréductible sur $\mathcal{F}$.

Remarque 4. *Un système irréductible est complètement réductible. Un système à la fois réductible et complètement réductible est décomposable.*

Notons que $[A]$ décomposable, réductible, complètement réductible au sens où nous l'avons défini ci-dessus est équivalent à $(\mathcal{F}^n, \Delta_A)$ décomposable, réductible, complètement réductible au sens des $\mathcal{D}$-modules.

Le sujet de ce chapitre est la *factorisation* de systèmes différentiels linéaires. Définissons donc précisément ce que nous entendons par *factoriser* un système différentiel.

Définition 10. *Soit $(\mathcal{F}, \partial)$ un corps différentiel et $A \in \mathbb{M}_n(\mathcal{F})$.* Factoriser le système différentiel linéaire $[A]$ sur $\mathcal{F}$ *signifie soit trouver un système équivalent (sur $\mathcal{F}$) $[B]$ avec B diagonale ou triangulaire par blocs, soit prouver qu'un tel système équivalent n'existe pas. Autrement dit, cela signifie décider si $[A]$ est décomposable ou indécomposable, réductible ou irréductible, et dans le cas décomposable ou réductible, calculer les facteurs.*

L'intérêt de factoriser un système différentiel réside surtout dans le fait qu'une fois le système factorisé, le résoudre se ramène à résoudre des systèmes de dimensions plus petites : si $[A]$ est équivalent à $[B]$ avec

$$[B] = P[A] = [B^{[1]}] \bigoplus [B^{[2]}],$$

alors résoudre $[A]$ revient à résoudre séparément $[B^{[1]}]$ et $[B^{[2]}]$. De manière analogue, si $[A]$ est équivalent à $[B] = P[A]$ avec

$$B = \begin{pmatrix} B^{[1]} & B^{[3]} \\ 0 & B^{[2]} \end{pmatrix},$$

alors résoudre $[A]$ peut se faire « en cascade » de la manière suivante : on résout tout d'abord $\partial(Y_2) = B^{[2]}\, Y_2$, puis on reporte la solution Y_2 dans l'équation $\partial(Y_1) = B_1\, Y_1 + B_3\, Y_2$ que l'on résout en Y_1.

1.1.6 Systèmes différentiels en caractéristique p

À partir de maintenant et jusqu'à la section 1.8, $\mathbb{K}$ désigne un corps de caractéristique $p > 0$ satisfaisant $[\mathbb{K} : \mathbb{K}^{(p)}] = p$, où $K^{(p)} := \{f^p \mid f \in \mathbb{K}\}$. Le corps $\mathbb{K}$ est donc un espace vectoriel de dimension p sur $\mathbb{K}^{(p)}$; dans la suite nous nous fixons un $x \in \mathbb{K} \setminus \mathbb{K}^{(p)}$, et nous identifions $\mathbb{K}$ à $\bigoplus_{i=0}^{p-1} \mathbb{K}^{(p)}\, x^i$.

Munissons $\mathbb{K}$ de la dérivation $' := \frac{d}{dx}$ définie par

$$y' = y_1 + 2\, y_2\, x + \cdots + (p-1)\, y_{p-1}\, x^{p-2},$$

pour tout $y = y_0 + y_1\, x + \cdots + y_{p-1}\, x^{p-1}$ appartenant à $\mathbb{K}$.

Ceci fait de $(\mathbb{K}, ')$ un corps différentiel ayant comme corps des constantes $\mathrm{Const}(\mathbb{K}) = \mathbb{K}^{(p)}$.

Dans nos algorithmes, nous nous placerons toujours dans le cas où $\mathbb{K} = k(x)$ avec $k = \mathbb{F}_q$ et $q = p^r$ avec r entier naturel supérieur ou égal à 1 et p premier. Dans ce cas, nous avons toujours $\mathbb{K}^{(p)} = k(x^p)$.

Pour alléger les notations, nous posons

$$\mathcal{C} := \mathrm{Const}(\mathbb{K}) = \mathbb{K}^{(p)}.$$

Soit $[A]$ un système différentiel linéaire avec $A \in \mathbb{M}_n(\mathbb{K})$. L'opérateur différentiel $\Delta_A : \mathbb{K}^n \to \mathbb{K}^n$ associé à $[A]$ peut être vu comme un endomorphisme de $\mathcal{C}^{np}$. Nous verrons par la suite que ce simple fait s'avère très utile : par exemple, nous pouvons l'utiliser pour calculer de manière très simple les solutions rationnelles d'un système différentiel linéaire en caractéristique p.

Dans ce chapitre, nous nous intéressons uniquement à la factorisation de systèmes différentiels en caractéristique p. Cependant, nous gardons toujours en tête le fait que le sujet sous-jacent est la factorisation en caractéristique zéro : étant donné un système différentiel $[A]$ en caractéristique zéro (à coefficients dans un corps de nombres), pour presque tout nombre premier p, nous pouvons réduire les coefficients de A modulo p et associer ainsi à $[A]$ un système différentiel en caractéristique p (à coefficients dans un certain $\mathbb{F}_q$, $q = p^r$). En effet, il suffit que p soit tel que les coefficients de la matrice A puissent être réduits modulo p ce qui n'en exclut qu'un nombre fini. Pour des détails concernant la réduction modulo p d'un élément e appartenant à un corps de nombres, nous renvoyons à la section 2.1.3 du chapitre 2.

1.2 Solutions rationnelles

Nous allons voir ici que calculer les solutions rationnelles d'un système différentiel linéaire (ou d'une équation différentielle linéaire scalaire) à coefficients dans $k(x)$ est beaucoup plus facile (au moins théoriquement) en caractéristique p (*e.g.*, $k = \mathbb{F}_p$) qu'en caractéristique zéro (*e.g.*, $k = \mathbb{Q}$). En caractéristique zéro, l'algorithme classique procède comme suit (voir par exemple [Ba99] pour les systèmes et [Br92] pour les équations scalaires) :

- on calcule un multiple du dénominateur et le degré du numérateur d'éventuelles solutions rationnelles,
- on calcule les solutions polynomiales (voir Chapitre 4) d'un système (ou opérateur) auxiliaire.

En caractéristique p, tout ces calculs se ramènent à la résolution d'un système linéaire homogène.

1.2.1 Cas d'un système

Soit $[A]$ un système différentiel linéaire avec $A \in \mathbb{M}_n(\mathbb{K})$. Dans cette section, nous nous intéressons au problème de trouver toutes les solutions rationnelles de $[A]$. Rappelons que l'ensemble des solutions rationnelles de $[A]$ est un espace vectoriel de dimension inférieure ou égale à n sur $\mathcal{C}$: nous allons chercher à en exhiber une base. Comme nous pouvons ici regarder Δ_A comme un endomorphisme de $\mathcal{C}^{np}$, le calcul d'une base de solutions rationnelles de $[A]$ se ramène à la résolution d'un système linéaire homogène de taille np.

En pratique, pour calculer les solutions rationnelles de $[A]$, il nous suffit donc de choisir une base de $\mathcal{C}^{np}$, former la matrice de Δ_A dans cette base et en calculer le noyau. Ceci peut se faire en procédant ainsi : soit $Y \in \mathbb{K}^n$; on peut écrire

$$Y = Y_0 + Y_1\, x + \cdots + Y_{p-1}\, x^{p-1},$$

où les Y_i appartiennent à $\mathcal{C}^n$. De la même façon, on a

$$A = A^{[0]} + A^{[1]}\, x + \cdots + A^{[p-1]}\, x^{p-1},$$

où les $A^{[i]}$ appartiennent à $\mathbb{M}_n(\mathcal{C})$.
On obtient alors

$$Y' = Y_1 + 2\, Y_2\, x + \cdots + (p-1)\, Y_{p-1}\, x^{p-2},$$

d'un côté, et de l'autre

$$A\, Y = C_0 + C_1\ x + \cdots + C_{p-1}\, x^{p-1},$$

avec

$$\forall\, h \in \{0, \ldots, p-2\},\ C_h = \sum_{i+j=h} A^{[i]}\, Y_j + x^p \sum_{i+j=h+p} A^{[i]}\, Y_j \text{ et } C_{p-1} = \sum_{i+j=p-1} A^{[i]}\, Y_j.$$

Alors $Y \in \mathbb{K}^n$ est solution de $Y' = A\,Y$ si, et seulement si,

$$\left\{\begin{array}{ccc} Y_1 & = & A^{[0]}\,Y_0 + x^p\,(A^{[1]}\,Y_{p-1} + \cdots + A^{[p-1]}\,Y_1), \\ 2\,Y_2 & = & A^{[1]}\,Y_0 + A^{[0]}\,Y_1 + x^p\,(A^{[2]}\,Y_{p-1} + \cdots + A^{[p-1]}\,Y_2), \\ & \vdots & \\ (p-1)\,Y_{p-1} & = & A^{[p-2]}\,Y_0 + A^{[p-3]}\,Y_1 + \cdots + A^{[0]}\,Y_{p-2} + x^p\,(A^{[p-1]}\,Y_{p-1}), \\ 0 & = & A^{[p-1]}\,Y_0 + A^{[p-2]}\,Y_1 + \cdots + A^{[0]}\,Y_{p-1}. \end{array}\right.$$

Ce qui, en écriture matricielle, nous donne $\mathcal{N}\,\tilde{Y} = 0$ où $\tilde{Y} = \begin{pmatrix} Y_0 \\ \vdots \\ Y_{p-1} \end{pmatrix} \in \mathcal{C}^{np}$ et

$$\mathcal{N} = \begin{pmatrix} A^{[0]} & x^p\,A^{[p-1]} - I_n & x^p\,A^{[p-2]} & \cdots & x^p\,A^{[1]} \\ A^{[1]} & \ddots & x^p\,A^{[p-1]} - 2\,I_n & \ddots & \vdots \\ & \ddots & \ddots & \ddots & x^p\,A^{[p-2]} \\ \vdots & & & \ddots & x^p\,A^{[p-1]} - (p-1)\,I_n \\ A^{[p-1]} & \cdots & \cdots & A^{[1]} & A^{[0]} \end{pmatrix}.$$

(I_n est la matrice identité de taille n).

Lorsque l'on cherche à calculer les solutions rationnelles d'un système différentiel $\frac{d}{dx}(Y) = A\,Y$ avec $A \in \mathbb{M}_n(k(x))$, alors on peut toujours se ramener au cas où A est à coefficients polynomiaux (dans $k[x]$) ; si D est le dénominateur commun des coefficients de A, alors il nous suffit de remplacer la dérivation $\frac{d}{dx}$ par $D\,\frac{d}{dx}$ et la matrice A par $D\,A \in \mathbb{M}_n(k[x])$.

Proposition 2. *Soit $[A]$ un système différentiel linéaire avec $A \in \mathbb{M}_n(k(x))$ où $k = \mathbb{F}_q$ avec $q = p^r$. Soit D le plus petit dénominateur commun des coefficients de A et $\tilde{A} := D\,A \in \mathbb{M}_n(k[x])$. Soit a une borne sur le degré (en x) des entrées de $\tilde{A}$. Soit ω un réel tel que le produit de deux matrices dans $\mathbb{M}_n(\mathcal{F})$, avec $\mathcal{F}$ corps quelconque, puisse se faire en $\mathcal{O}(n^\omega)$ produits dans $\mathcal{F}$. On peut calculer une base de solutions rationnelles de $[A]$ en $\mathcal{O}(n^\omega\,\max(a,p)^2\,p^{\omega-2})$ opérations dans k .*

Démonstration. L'explication précédant la proposition montre qu'il nous suffit de calculer le noyau de la matrice $\tilde{\mathcal{N}}$ obtenue en appliquant le calcul fait ci-dessus au système $D\,\frac{d}{dx}(Y) = \tilde{A}\,Y$. D'après [MS03, Theorem 6.1], le noyau d'une matrice polynomiale de dimensions $n \times m$ (avec $n \geq m$) dont le degré des entrées est borné par a peut se calculer en $\mathcal{O}(n^\omega\,a^2)$ opérations de corps. Soit $a_{i,j} \in k[x]$ les entrées de $\tilde{A}$. Pour tout i, j, on peut écrire $a_{i,j} = \sum_{h=0}^{p-1} a_{i,j}^{[h]}\,x^h$ avec les $a_{i,j}^{[h]}$ dans $k[x^p]$. Le degré (en x) des $a_{i,j}$ étant borné

par a, celui (en x^p) des entrées $a_{i,j}^{[h]}$ de $\tilde{A}^{[h]}$ est borné par a/p. La matrice $\tilde{\mathcal{N}}$ possède donc des entrées de degré (en x^p) borné par $\lceil a/p \rceil$ et est de dimension $(n\,p) \times (n\,p)$. Par conséquent, si $a \geq p$, on peut considérer les entrées de $\tilde{\mathcal{N}}$ comme des polynômes en la variable $c := x^p$ de degré borné par $a/p + 1$; dans ce cas, son noyau se calcule en $\mathcal{O}(n^\omega\, a^2\, p^{\omega-2})$ opérations dans k. Si $a < p$ (ce qui est équivalent au fait que les $a_{i,j}^{[k]}$ sont dans k) et si A n'est pas à coefficients dans k, le degré en x^p des entrées de $\tilde{\mathcal{N}}$ est 1 et la complexité du calcul du noyau de $\tilde{\mathcal{N}}$ devient $\mathcal{O}(n^\omega\, p^\omega)$ opérations dans k. Finalement, si A est à coefficients dans k, alors les entrées de $\tilde{\mathcal{N}}$ appartiennent à k et son noyau peut aussi se calculer en $\mathcal{O}(n^\omega\, p^\omega)$ opérations dans k. □

Exemple 2. *Prenons $\mathbb{K} := \mathbb{F}_p(x)$ avec $p = 3$ et considérons le système différentiel donné par la matrice suivante :*

$$A := \begin{bmatrix} -\dfrac{x^2+1+x^3-x}{x\,(x^2+1)} & \dfrac{2+x^3-x}{x\,(x^2+1)} \\ \dfrac{x\,(x^2-1)}{x^2+1} & -\dfrac{-x^2+1+x^3-x}{x^2+1} \end{bmatrix}.$$

Pour cet opérateur, la matrice $\mathcal{N}$ définie ci-dessus est la suivante (c est la constante x^p) :

$$\mathcal{N} = \begin{bmatrix} \frac{2\,c^2+1}{1+c^2} & \frac{c^2+2\,c+2}{1+c^2} & \frac{c\,(c+2+2\,c^2)}{c+c^3}+2 & \frac{c\,(2\,c+2)}{c+c^3} & \frac{2\,c^2}{1+c^2} & \frac{c\,(c+1)}{1+c^2} \\ \frac{2\,c}{1+c^2} & \frac{c+2+c^2}{1+c^2} & \frac{c^2}{1+c^2} & \frac{c\,(2\,c+2)}{1+c^2}+2 & \frac{c\,(c^2+2)}{1+c^2} & \frac{c\,(2\,c^2+c+1)}{1+c^2} \\ \frac{2\,c}{1+c^2} & \frac{c+1}{1+c^2} & \frac{2\,c^2+1}{1+c^2} & \frac{c^2+2\,c+2}{1+c^2} & \frac{c\,(c+2+2\,c^2)}{c+c^3}+1 & \frac{c\,(2\,c+2)}{c+c^3} \\ \frac{c^2+2}{1+c^2} & \frac{2\,c^2+c+1}{1+c^2} & \frac{2\,c}{1+c^2} & \frac{c+2+c^2}{1+c^2} & \frac{c^2}{1+c^2} & \frac{c\,(2\,c+2)}{1+c^2}+1 \\ \frac{c+2+2\,c^2}{c+c^3} & \frac{2\,c+2}{c+c^3} & \frac{2\,c}{1+c^2} & \frac{c+1}{1+c^2} & \frac{2\,c^2+1}{1+c^2} & \frac{c^2+2\,c+2}{1+c^2} \\ \frac{c}{1+c^2} & \frac{2\,c+2}{1+c^2} & \frac{c^2+2}{1+c^2} & \frac{2\,c^2+c+1}{1+c^2} & \frac{2\,c}{1+c^2} & \frac{c+2+c^2}{1+c^2} \end{bmatrix}.$$

En utilisant notre procédure rat-sol2 *(qui après avoir calculé la matrice $\mathcal{N}$ ci-dessus calcule son noyau et en déduit une base de solutions rationnelles), on obtient la base de solutions rationnelles suivante :*

```
> y_2:=rat_sol2(A,2,3);
```

$$y_2 := [2\,c + x^2,\ 2\,c]$$

1.2.2 Cas d'un opérateur scalaire

Appliquons maintenant la même stratégie pour calculer les solutions rationnelles d'un opérateur différentiel linéaire scalaire $L := \sum_{i=0}^{n} a_i\,(d/dx)^i$ (avec les a_i dans $k[x]$) vu comme une application $\mathcal{C}(x^p)$-linéaire de $k[x]$ sur $k[x]$. On forme donc la matrice de cette application linéaire en évaluant les $L(x^i)$ pour $i = 0, \ldots, p-1$ et en les décomposant sur la base $1, x, \ldots, x^{p-1}$ de $k(x)$ sur $k(x^p)$. Soit l une borne sur le degré (en x) des coefficients a_i de L. Alors une borne sur le degré en x^p des $L(x^i)$ à considérer est $l + p$.

Par conséquent, la matrice dont nous devons calculer le noyau est de dimension $p \times p$ et admet $l/p+1$ comme borne sur le degré en x^p de ses entrées. Une analyse analogue à celle de la preuve de la proposition 2 nous fournit donc une complexité en $\mathcal{O}(\max(l,p)^2\, p^{\omega-2})$ opérations dans k.

Par conséquent, on peut penser que calculer les solutions rationnelles d'un système différentiel linéaire $[A]$ en caractéristique p en le transformant en une équation scalaire L à l'aide d'un vecteur cyclique pourrait être plus efficace. On sait, d'après la proposition 1, qu'en caractéristique p (avec $p > n$), calculer un vecteur cyclique Λ_0 coûte $\mathcal{O}(n^5)$ opérations dans k. De plus, si les degrés de Λ et A sont respectivement bornés par λ et a, alors on obtient une borne de la forme $n\lambda + a\,\frac{n\,(n+1)}{2}$ pour le degré des coefficients de L (voir Proposition 1). Supposons maintenant que $p > a$ et que p est aussi plus grand que le degré des coefficients du vecteur cyclique (par exemple $p > n\,\lambda + a\,\frac{n\,(n+1)}{2}$). Dans ce cas, calculer les solutions rationnelles en passant par un vecteur cyclique coûte $\mathcal{O}(n^5 + p^\omega)$ opérations dans k alors qu'un calcul direct sur le système coûte $\mathcal{O}(n^\omega\, p^\omega)$. Par conséquent, nous pouvons conclure que la méthode du vecteur cyclique est plus efficace seulement lorsque p est beaucoup plus grand que n. Lorsque p et n ont le même ordre de grandeur (ou lorsque p est plus petit que n et qu'un vecteur cyclique existe), alors un calcul direct est plus efficace.

1.3 La p-courbure d'un système différentiel linéaire

1.3.1 Définition et calcul

Définition 11. *Soit $[A]$ un système différentiel linéaire avec $A \in \mathbb{M}_n(\mathbb{K})$. La* p-courbure *de $[A]$ est l'opérateur de $\mathbb{K}^n$ dans $\mathbb{K}^n$ défini par*

$$\Delta_A^p := \left(\frac{d}{dx} - A\right)^p.$$

Avant de montrer comment calculer cet objet en pratique, nous donnons deux premières propriétés ainsi que quelques exemples.

Lemme 1. *Soit $[A]$ un système différentiel avec $A \in \mathbb{M}_n(\mathbb{K})$. La p-courbure Δ_A^p du système $[A]$ est une application $\mathbb{K}$-linéaire.*

Démonstration. On montre tout d'abord par récurrence sur j que pour tout $j \geq 0$ et pour tout couple (a, v) dans $\mathbb{K} \times \mathbb{K}^n$,

$$\Delta_A^j(a\,v) = \sum_{i=0}^{j} \binom{j}{i}\, a^{(i)} \Delta_A^{j-i}(v).$$

Par conséquent, pour tout couple (a, v) de $\mathbb{K} \times \mathbb{K}^n$,

$$\Delta_A^p(a\,v) = a^{(p)}\, v + a\,\Delta_A^p(v),$$

et le fait que l'application $(\frac{d}{dx})^p$ soit nulle sur $\mathbb{K}$ (ce qui implique $a^{(p)} = 0$ dans la précédente formule) nous permet de conclure. □

Lemme 2. *Si deux systèmes différentiels linéaires* $[A]$ *et* $[B]$ *sont équivalents, alors leurs p-courbures sont semblables. Précisément, si* $[B] = P[A]$*, alors* $\Delta_B^p = P^{-1}\,\Delta_A^p\,P$.

Démonstration. Ceci provient directement de la remarque que si $B = P[A]$, alors $\Delta_B = P^{-1}\,\Delta_A\,P$. □

Exemple 3. *On peut vérifier que :*
(a). La p-courbure du système différentiel $y' = \mu\,y$*, de dimension* 1*, est la multiplication par* $\mu^{(p-1)} + \mu^p$.
(b). Si $A = \mu\,I$*, alors* $\Delta_A^p = (\mu^{(p-1)} + \mu^p)\,I$.
(c). Si $A = \bigoplus_{i=1}^r \mu_i\,I_{n_i}$*, alors* $\Delta_A^p = \bigoplus_{i=1}^r (\mu_i^{(p-1)} + \mu_i^p)\,I_{n_i}$.
(d). Si $A = \bigoplus_{i=1}^r A^{[i]}$*, alors* $\Delta_A^p = \bigoplus_{i=1}^r \Delta_{A^{[i]}}^p$.
(e). Si $[A^*]$ *représente le système adjoint de* $[A]$*, alors* $\Delta_{A^*}^p = -{}^t\Delta_A^p$*. Autrement dit, la p-courbure de l'adjoint (au sens des systèmes différentiels) est égale à l'adjoint de la p-courbure en appelant* adjoint d'une matrice M *la matrice* $-{}^tM$.

Dans [Ka82], Katz propose un algorithme pour le calcul de la p-courbure d'un système différentiel : on définit la suite $(A_i)_{i\geq 0}$, que nous appelons *suite de Lie de* $[A]$, par

$$A_0 := I_n \;\text{ et }\; \forall i \geq 0,\; A_{i+1} := \Delta_A(A_i) = A_i' - A\,A_i. \tag{1.1}$$

On calcule A_2, $A_3, \ldots, A_p$ et A_p est exactement la matrice de l'opérateur différentiel Δ_A^p dans la base canonique de $\mathbb{K}^n$, *i. e.*, la p-courbure. Dans la suite, nous ne ferons pas de distinction entre l'opérateur Δ_A^p et sa matrice A_p.

Soit $A \in \mathbb{M}_n(k(x))$ avec $k = \mathbb{F}_q$ où $q = p^r$. En pratique, pour calculer A_p, nous adaptons les formules (1.1) pour que tous les calculs se fassent dans $k[x]$, *i. e.*, sans dénominateurs (voir la procédure LiePol dans l'annexe) ; soit $D \in k[x]$ tel que $\tilde{A} := D\,A \in \mathbb{M}_n(k[x])$. Alors, pour tout $i \geq 1$, $\tilde{A}_i := D^i\,A_i \in \mathbb{M}_n(k[x])$ et on obtient ainsi la formule de récurrence suivante :

$$\tilde{A}_{i+1} = D\,\tilde{A}_i' - (D'\,i + \tilde{A})\,\tilde{A}_i. \tag{1.2}$$

Proposition 3. *Soit* $[A]$ *un système différentiel avec* $A \in \mathbb{M}_n(k(x))$ *où* $k = \mathbb{F}_q$*,* $q = p^r$*. Soit* D *le plus petit dénominateur commun des coefficients de* A *et* $\tilde{A} := D\,A \in \mathbb{M}_n(k[x])$*. Soit* a *une borne sur le degré (en* x*) des entrées de* $\tilde{A}$*. Soit* ω *un réel tel que le produit de deux matrices dans* $\mathbb{M}_n(\mathcal{F})$*, avec* $\mathcal{F}$ *corps quelconque, puisse se faire en* $\mathcal{O}(n^\omega)$ *produits dans* $\mathcal{F}$*. Alors, la p-courbure de* $[A]$ *peut se calculer en* $\mathcal{O}(n^\omega\,a\,\log(a)\,p^2)$ *opérations dans* k.

Précisons que, par exemple, la méthode de Strassen pour la multiplication de matrices correspond à $\omega = \log_2 7$ (voir [GG99, 12.1]).

Démonstration. D'après (1.2), calculer la p-courbure nécessite p produits de matrices de polynômes de dimension $n \times n$ ayant des coefficients de degré respectivement bornés par $a, 2a, \ldots, (p-1)\,a$. Ceci peut donc se faire en $\mathcal{O}(n^\omega\,p)$ produits de polynômes. Maintenant, on peut conclure car la F.F.T. ([GG99, Section 8]) permet de calculer le produit de deux polynômes de degrés respectivement bornés par a et $i\,a$ en $\mathcal{O}(i\,a\,\log(a))$ opérations de corps. □

Calculer la p-courbure (pour $p > n$) d'un opérateur différentiel scalaire ayant des coefficients de degré borné par l en utilisant les formules de récurrences données dans [PS03, 13.2.2, p.335] donne une complexité en $\mathcal{O}(n^2\, l\, \log(l)\, p)$ opérations dans k. Ici aussi comparons le calcul direct de la p-courbure du système à la méthode passant par un vecteur cyclique et, pour simplifier, restreignons nous au cas où le système est à coefficients polynomiaux, *i. e.*, $A \in \mathbb{M}_n(k[x])$ où $k = \mathbb{F}_q$, $q = p^r$. D'après la proposition 1, si a est une borne sur le degré des entrées de A, une borne sur le degré des coefficients de l'opérateur L calculé à partir d'un vecteur cyclique est $b = n\lambda + a\frac{(n+1)n}{2}$ (λ borne sur le degré des coefficients du vecteur cyclique). Donc, *via* un vecteur cyclique, la p-courbure se calcule en $\mathcal{O}(n^5 + n^2\, b\, \log(b)\, p)$ opérations dans k. Par conséquent, le calcul en passant par un vecteur cyclique est plus avantageux seulement lorsque p est beaucoup plus grand que n. Dans le cas contraire où p et n ont le même ordre de grandeur, le calcul direct sur le système est plus efficace. Rappelons que le cas $p < n$ est exclu car l'existence d'un vecteur cyclique n'est alors plus assurée.

1.3.2 Classification des modules différentiels

Dans [Pu95] (voir aussi [PS03, Chapitre 13]), van der Put donne une classification des modules différentiels en caractéristique p. La conséquence de cette classification pour le problème de factorisation est la suivante : la forme de Jordan (sur $\overline{\mathcal{C}}$) de la p-courbure d'un système différentiel linéaire $[A]$ donne toutes les factorisations de $[A]$. Voici brièvement comment est construite cette classification (pour plus de détails, voir [Pu95] ou [PS03, Ch. 13]) : considérons les deux catégories suivantes :

- les modules différentiels (voir Définition 3) munis du produit tensoriel usuel,
- les $\mathcal{C}[T]$-modules ($\mathcal{C}[T]$: anneau commutatif des polynômes à coefficients dans $\mathcal{C}$) de dimension finie sur $\mathcal{C}$. Un élément de cette catégorie est donné par un couple (N, Θ) où N est un $\mathbb{K}$-espace vectoriel de dimension n, Θ est un endomorphisme de N et l'action de $\mathcal{C}[T]$ sur N est définie par

$$f.m = f(\Theta)\, m,\ \forall f \in \mathcal{C}[T],\ \forall m \in N,$$

  et cette seconde catégorie est munie du produit tensoriel défini par $(N_1, \Theta_1) \otimes_{\mathcal{C}} (N_2, \Theta_2) = (N_1 \otimes_{\mathcal{C}} N_2, \Theta)$ où $\Theta = \Theta_1 \otimes_{\mathcal{C}} I_{N_1} + I_{N_2} \otimes_{\mathcal{C}} \Theta_2$.

Le résultat de van der Put peut se résumer ainsi : si le corps $\mathbb{K}$ possède une certaine propriété[1], par exemple $\mathbb{K} = \overline{\mathbb{F}}_p(x)$ ou $\overline{\mathbb{F}}_p((x))$, alors le foncteur qui envoie $(\mathcal{C}^n, \Theta)$ sur $(\mathbb{K}^n, \Delta)$ où Δ est déterminé tel que $\Delta^p = I \otimes_{\mathcal{C}} \Theta$ (avec l'identification $\mathbb{K}^n = \mathbb{K} \otimes_{\mathcal{C}} \mathcal{C}^n$) fournit l'équivalence entre les deux catégories décrites ci-dessus. La forme de Jordan (sur $\overline{\mathcal{C}}$) de la p-courbure Δ_A^p donne la factorisation complète du $\mathcal{C}[T]$-module $(\mathcal{C}^n, \Theta_A)$ avec $\Delta_A^p = I \otimes_{\mathcal{C}} \Theta_A$. Par conséquent, comme le foncteur défini ci-dessus respecte les constructions de l'algèbre linéaire, ceci nous donne aussi la factorisation complète du

[1] Il n'existe pas de corps non-commutatif de degré p^2 sur son centre Z avec Z extension finie de $\mathbb{K}$. C'est le cas lorsque $\mathbb{K}$ est un *corps* C_1 ([Ja80, Definition 11.5, p. 649]), ou *corps quasi-algébriquement clos*.

module différentiel $(\mathbb{K}^n, \Delta_A)$ associé.

Dans la suite, nous allons expliciter comment utiliser cette classification pour factoriser effectivement des systèmes différentiels en caractéristique p. Une manière de rendre cette classification algorithmique est développée dans [Pu97] (certains éléments sont donnés dans [PS03, Ch. 13]) pour le cas d'une équation différentielle scalaire. Ici nous nous intéressons aux systèmes différentiels et nous proposons certaines variantes (par exemple l'utilisation de l'eigenring comme dans [Ba01]). Certains résultats pouvant se voir à partir de cette classification sont redémontrés d'une manière élémentaire : ceci nous permet d'exhiber des preuves constructives nécessaires pour implanter un algorithme de factorisation de systèmes différentiels en caractéristique p. De plus, le fait d'écrire ces preuves nous permet de voir comment améliorer certains résultats (voir Théorème 1 dans la sous-section 1.3.4) et s'avère, par la suite, indispensable pour généraliser notre algorithme à d'autres cas et notamment à celui des systèmes d'équations aux dérivées partielles où la classification ne s'applique plus directement.

1.3.3 Systèmes ayant une p-courbure nulle

Comme l'illustre le paragraphe précédent, la notion de p-courbure est fondamentale dans la théorie des équations différentielles en caractéristique p. Ceci s'explique aussi par son lien avec la conjecture de Grothendieck que nous rappelons ici. Dans le cas où le système est à coefficients dans $\mathbb{Q}(x)$, cette conjecture peut s'énoncer de la manière suivante :

Conjecture 1 (Grothendieck). *Soit $[A]$ un système différentiel linéaire avec $A \in \mathbb{M}_n(\mathbb{Q}(x))$. Alors, $[A]$ possède une base de solutions algébriques si, et seulement si, pour presque tout p, $[A[p]]$ possède une base de solutions rationnelles où $[A[p]]$ désigne le système réduit modulo p.*

L'implication qui n'a toujours pas été prouvée est « $\Leftarrow$ ». Il existe cependant des preuves de cette conjecture dans beaucoup de cas : pour un historique de cette conjecture, voir [Ch01] ou [Pu01]. Dans [Ka82], Katz propose une conjecture équivalente en termes d'algèbres de Lie. Notons enfin que la conjecture analogue dans le cas des équations aux q-différences a été démontrée par Di Vizio dans [DiV02].

Le lemme suivant dû à Cartier (voir [Pu96, Lemma 2.1] ou encore [Ka70, Theorem 5.1] dans un cadre plus général) fait le lien entre la conjecture de Grothendieck et la p-courbure.

Lemme 3 (Cartier). *Soit $[A]$ un système différentiel avec $A \in \mathbb{M}_n(\mathbb{K})$. Le système différentiel $[A]$ a une base de solutions rationnelles si, et seulement si, sa p-courbure A_p est nulle.*

Ceci implique que si un système $[A]$ possède une p-courbure nulle, alors $[A]$ est équivalent sur $\mathbb{K}$ au système nul $[0]$. En effet, il suffit de prendre pour matrice de passage P une matrice fondamentale de solutions rationnelles de $[A]$ et on a $P[A] = [0]$.

Exemple 4. *Prenons* $\mathbb{K} := \mathbb{F}_p(x)$ *avec* $p = 5$ *et considérons le système* $Y' = AY$ *de dimension* $n = 2$ *donné par la matrice suivante :*

$$A := \begin{bmatrix} 0 & 1 \\ \dfrac{3\,x}{x^3+1} & \dfrac{3+x^3}{x^4+x} \end{bmatrix}.$$

On en calcule la p-courbure avec notre procédure LiePol*, qui suit notre méthode calculant sans dénominateurs.*

```
> pcurv := LiePol(A,n,p);
```

$$pcurv := \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

La p-courbure étant nulle, nous savons que notre système a une base de solutions rationnelles. Utilisons donc la procédure rat-sol2 *et formons la matrice ayant pour colonnes les éléments de cette base de solutions rationnelles.*

```
> P:=concat(rat_sol2(A,n,p));
```

$$P := \begin{bmatrix} x^4 & 4+x^3 \\ 4\,x^3 & 3\,x^2 \end{bmatrix}$$

On sait alors que cette matrice satisfait $P[A] = [0]$.

Pour être complet, nous rappelons ici un autre résultat élémentaire concernant les systèmes ayant une p-courbure nilpotente et nous renvoyons à [CC85] pour une étude plus poussée de ce cas.

Lemme 4. *Soit* $[A]$ *un système différentiel linéaire avec* $A \in \mathbb{M}_n(\mathbb{K})$*. Si la p-courbure de* $[A]$ *est nilpotente, alors* $[A]$ *possède au moins une solution rationnelle (non-nulle).*

Démonstration. Supposons que la p-courbure soit nilpotente d'ordre r, *i. e.*, $\Delta_A^{pr} = 0$ et $\Delta_A^{p(r-1)} \neq 0$. Alors, il existe $i \in \{p\,(r-1), \ldots, p\,r\}$ tel que $\Delta^i = 0$ et $\Delta^{i-1} \neq 0$. Il existe donc $e \in \mathbb{K}^n$ tel que $\Delta^i(e) = 0$ et $f := \Delta_A^{i-1}(e) \neq 0$. Par conséquent, f est un élément non-nul du noyau de Δ_A. □

1.3.4 Systèmes ayant une p-courbure scalaire

Définition 12. *Soit* $[A]$ *un système différentiel linéaire avec* $A \in \mathbb{M}_n(\mathbb{K})$*. On dit que* $[A]$ a une p-courbure scalaire *si* $A_p = \lambda\, I_n$ *avec* $\lambda \in \mathcal{C}$.

Notons qu'en toute généralité, λ devrait appartenir à $\mathbb{K}$ mais ici le fait que Δ_A commute avec Δ_A^p implique que λ appartient nécessairement à $\mathcal{C}$.

Il est connu que si la p-courbure est scalaire alors le système est équivalent sur $\overline{\mathbb{K}}$ à un système sous forme diagonale. Nous énonçons ici quelques lemmes techniques

qui mènent à ce résultat et nous montrons comment adapter les preuves pour obtenir une équivalence non plus sur $\overline{\mathbb{K}}$ mais sur $\mathbb{K}$. Ce résultat de rationnalité est utile non seulement en pratique mais aussi en théorie : en effet, nous verrons dans la section 1.8 que ce résultat est indispensable pour étendre notre algorithme au cas des systèmes d'équations aux dérivées partielles où la classification de van der Put n'est plus applicable directement.

Lemme 5. *Soit A et B deux matrices telles que pour tout i, $j \in \mathbb{N}$, $A^{(i)}$ et $B^{(j)}$ commutent où $A^{(i)}$ désigne la i-ème dérivée de la matrice A. Si l'on pose $S = A + B$, on a alors : pour tout $i \in \mathbb{N}$,*

$$S_i = \sum_{k=0}^{i} \binom{i}{k} A_k \, B_{i-k},$$

où $(S_i)_i$, $(A_i)_i$ et $(B_i)_i$ représentent respectivement les suites de Lie de S, A et B (définies par les formules (1.1)). En particulier, pour $i = p$, on obtient

$$S_p = A_p + B_p.$$

Démonstration. Il suffit de procéder par récurrence sur i· □

Ce lemme s'applique en particulier lorsque $S = A + \mu\, I_n$ avec $\mu \in \mathbb{K}$ et implique alors $S_p = A_p + (\mu^p + \mu^{(p-1)})\, I_n$.

Dans la suite de cette sous-section concernant le cas où la p-courbure est scalaire, $\mathbb{K} := k(x)$ où k désigne un certain $\mathbb{F}_q$ avec $q = p^r$.

Lemme 6. *Soit τ l'application définie par :*

$$\tau \,:\, k(x) \to k(x^p),\ y \mapsto y^p + y^{(p-1)}.$$

L'équation différentielle scalaire $\tau(y) = c$ avec $c \in k(x^p)$ possède toujours une solution dans $\overline{k}(x)$.

Des résultats portant sur cette équation $\tau(y) = c$ avec $c \in \mathcal{C}$ se trouvent dans [PS03, 13.2.1] et [Pu97, 4.2]. Cette propriété n'est donc pas nouvelle mais nous en donnons quand même une preuve ; il sera utile d'avoir cette preuve en tête lorsque nous étudierons le cas des systèmes d'équations aux dérivées partielles dans la section 1.8. Nous avons tout d'abord besoin du lemme suivant :

Lemme 7. *Soit $j \in \mathbb{Z}$ et p un nombre premier. Alors, on a :*

$$(x^j)^{(p-1)} = \begin{cases} -x^{j-p+1} & \text{si } j \equiv -1 \mod p, \\ 0 & \text{sinon.} \end{cases}$$

Démonstration. Nous savons que $(x^j)^{(m)} = j\,(j-1)\cdots(j-m+1)\,x^{j-m}$ (pour $m \geq 1$) et donc, $(x^j)^{(p-1)} = j\,(j-1)\cdots(j-p+2)\,x^{j-p+1}$. En caractéristique p, on a

$$j\,(j-1)\cdots(j-p+2) = \begin{cases} (p-1)! & \text{si } j \equiv -1 \mod p, \\ 0 & \text{sinon.} \end{cases}$$

Or, d'après le théorème de Wilson, $(p-1)! = -1$ et le lemme est donc démontré. □

Nous pouvons maintenant donner une démonstration du lemme 6.

Démonstration. Avec les notations de l'énoncé du lemme 6, $c \in k(x^p)$ peut s'écrire $f(x^p)$ où $f(x) \in k(x)$. Soit

$$f(x) = \sum_{i=1}^{d} \sum_{j=1}^{d_i} \frac{b_{ij}}{(x-\xi_i)^j} + \sum_{j=0}^{e} a_j\, x^j, \tag{1.3}$$

la décomposition en éléments simples de $f(x)$ sur $\overline{k}$. Recherchons une solution $y(x)$ sous la forme

$$y(x) = \sum_{i=1}^{d} \sum_{j=1}^{d_i} \frac{\beta_{ij}}{(x-\xi_i^{1/p})^j} + \sum_{j=0}^{e} \alpha_j\, x^j.$$

On a alors,

$$y(x)^{(p)} = \sum_{i=1}^{d} \sum_{j=1}^{d_i} \frac{\beta_{ij}^p}{(x^p-\xi_i)^j} + \sum_{j=0}^{e} \alpha_j^p\, x^{pj}, \tag{1.4}$$

et d'après le lemme 7,

$$y^{(p-1)}(x) = \sum_{i=1}^{d} \sum_{j\equiv 1[p]} \frac{-\beta_{ij}}{\left(x^p-\xi_i\right)^{\frac{j-1}{p}+1}} + \sum_{j\equiv -1[p]} -\alpha_j x^{p(\frac{j+1}{p}-1)}. \tag{1.5}$$

Maintenant, en utilisant les équations (1.3), (1.4), (1.5) et l'unicité de la décomposition en éléments simples, $y^p + y^{(p-1)} = c$ est équivalent à :

$$\left\{ \begin{array}{rcll} \alpha^p_{\frac{j+1}{p}-1} - \alpha_j & = & a_{\frac{j+1}{p}-1}, & \forall\, j \equiv -1 \mod p,\, j \leq p\,(e+1)-1, \\ \alpha_j & = & 0, & \forall\, j \geq e+1, \end{array} \right.$$

et

$$\left\{ \begin{array}{rcll} \beta^p_{i,\frac{j-1}{p}+1} - \beta_{ij} & = & b_{i,\frac{j-1}{p}+1}, & \forall\, j \equiv 1 \mod p,\, j \leq p\,(d_i-1)+1, \\ \beta_{ij} & = & 0, & \forall\, j \geq d_i+1. \end{array} \right.$$

Remarquons que toutes ces équations, en les α_i et β_{ij}, ont clairement une solution sur $\overline{k}$ (ce qui termine la preuve). Plus précisément, les α_i peuvent toujours être choisis dans k (car les a_i sont dans k) mais ce n'est pas le cas pour les β_{ij}. En effet, les équations avec $j = 1$ pour les β_{ij} s'écrivent $\beta_{i1}^p - \beta_{i1} = b_{i1}$ et donc, en général, les β_{i1} sont algébriques sur k. □

Lemme 8. *Soit $[A]$ un système différentiel avec $A \in \mathbb{M}_n(k(x))$. La p-courbure de $[A]$ est scalaire si, et seulement si $[A]$ est équivalent (sur $\overline{k}(x)$) à $[\mu\, I_n]$ avec $\mu \in \overline{k}(x)$.*

Démonstration. Supposons que $A_p = -\lambda\, I_n$ pour un certain $\lambda \in k(x^p)$. Soit $\mu \in \overline{k}(x)$ une solution rationnelle de l'équation $\tau(y) = \lambda$ (une telle solution existe d'après le lemme 6). Alors, comme nous l'avons déjà remarqué, en posant $B = A + \mu\, I_n \in \mathbb{M}_n(\overline{k}(x))$, le lemme 5 nous donne $B_p = A_p + \lambda\, I_n = 0$. Donc, le système $[B]$ possède une p-courbure nulle et est équivalent (sur $\overline{k}(x)$) au système $[0]$. Par conséquent, il existe $P \in \mathrm{GL}_n(\overline{k}(x))$ tel que $P^{-1}\,(B\,P - P') = 0$ et en remplaçant B par $A + \mu\, I_n$ dans cette égalité, il vient $P^{-1}\,(A\,P - P') = -\mu\, I_n$. La réciproque étant évidente, le résultat est prouvé. □

Nous donnons maintenant une version rationnelle du lemme 8 qui constitue notre contribution dans ce cas où la p-courbure est scalaire :

Théorème 1. *Soit $[A]$ un système différentiel avec $A \in \mathbb{M}_n(k(x))$. Si p ne divise pas n, la p-courbure de $[A]$ est scalaire si, et seulement si, $[A]$ est équivalent (sur $k(x)$) à $[\mu\, I_n]$ avec $\mu \in k(x)$.*

Démonstration. Nous avons $A_p = -\lambda\, I_n$ et nous savons d'après le lemme précédent qu'il existe $P \in \mathrm{GL}_n(\overline{k}(x))$ tel que $P[A] = P^{-1}\, A\, P - P^{-1}\, P' = -\mu\, I_n$ avec $\mu \in \overline{k}(x)$ et $\tau(\mu) = \lambda$. Donc $\mathrm{tr}(A) - \frac{\det(P)'}{\det(P)} = -n\,\mu$ et pour p ne divisant n, nous avons $\mu = -\frac{1}{n}\,\mathrm{tr}(A) + \frac{1}{n}\,\frac{\det(P)'}{\det(P)}$; ceci implique que $-\frac{1}{n}\,\mathrm{tr}(A) \in k(x)$ est une solution de $\tau(y) = \lambda$. Alors, en reprenant la démonstration du lemme précédent et en remplaçant $\mu \in \overline{k}(x)$ par $-\frac{1}{n}\,\mathrm{tr}(A) \in k(x)$, on obtient le résultat annoncé. □

Ce résultat de rationnalité donne une méthode rationnelle et algorithmique pour factoriser (sur $k(x)$) des systèmes qui possèdent une p-courbure scalaire. Une astuce analogue pour calculer à partir d'un opérateur différentiel scalaire $L \in k(x)[\partial]$ ayant une p-courbure de la forme $\lambda\, I$ avec $\lambda \in k(x^p)$ une solution $\mu \in k(x)$ de l'équation $\tau(y) = \lambda$ est donnée dans [Pu97, 2.11]. La méthode est la suivante : soit $M = \mathrm{gcrd}(L, \partial^p - \lambda) = \partial^m + b_{m-1}\,\partial^{m-1} + \cdots + b_0$ où $m \in \mathbb{N}$ et gcrd représente le plus grand commun diviseur à droite d'opérateurs différentiels. L'élément $-\frac{b_{m-1}}{m}$ est alors solution de $\tau(y) = \lambda$.

Exemple 5. *Si l'on essaye de résoudre l'équation $\tau(y) = 1/x^5$ dans $\mathbb{F}_p(x)$ pour $p = 5$, alors en suivant la preuve du lemme 6, on trouve que y doit s'écrire β/x où β est solution de $\beta^5 - \beta = 1$. Cette équation n'ayant pas de solutions dans $\mathbb{F}_p$, l'équation $\tau(y) = 1/x^5$ ne possède pas de solutions dans $\mathbb{F}_p(x)$ mais seulement dans $\overline{\mathbb{F}}_p(x)$. D'après le théorème précédent, il n'existe donc pas de systèmes différentiels à coefficients dans $\mathbb{F}_5(x)$ et de dimension $n \neq 5\,n_1$ dont la p-courbure est égale à $-1/x^5\, I_n$. Ceci illustre le fait que la p-courbure d'un système différentiel à coefficients dans $k(x)$ ne peut pas être n'importe quoi.*

1.3.5 Solutions rationnelles et p-courbure

Nous rappelons maintenant un résultat, dû à Katz (voir [Ka70]), faisant la liaison entre l'espace $\mathcal{S}_A$ des solutions rationnelles d'un système différentiel et le noyau de sa p-courbure. Nous reproduisons ici la preuve de Katz non seulement pour la convenance du lecteur mais aussi pour son caractère algorithmique.

Soit $\mathcal{S}_A := \{Y \in \mathbb{K}^n \mid Y' = A\, Y\}$ l'espace des solutions rationnelles de $[A]$. Il est clair que $\mathcal{S}_A$ est inclus dans le noyau $\ker(A_p)$ de la p-courbure A_p (car $\Delta_A(Y) = 0$ implique $\Delta_A^p(Y) = 0$).

Lemme 9 (Katz). $\ker(A_p) = \mathcal{S}_A \otimes_{\mathcal{C}} \mathbb{K}$, *et en particulier,* $\dim_{\mathcal{C}} \mathcal{S}_A = \dim_{\mathbb{K}} \ker(A_p)$.

Démonstration. La projection

$$\mathrm{P_r} \,:\, \mathbb{K}^n \to \mathbb{K}^n,\ v \mapsto \mathrm{P_r}(v) := \sum_{i=0}^{p-1} \frac{(-x)^i}{i!}\, \Delta_A^i(v)$$

envoie $\ker(A_p)$ sur $\mathcal{S}_A$. En effet, on vérifie facilement que $\forall v \in \mathbb{K}^n$, $\Delta_A(\mathrm{Pr}(v)) = -(-x)^{p-1}\,\Delta_A^p(v)$. Par conséquent, si v appartient à $\ker(A_p)$, comme tous les $\Delta_A^i(v)$ appartiennent alors à $\ker(A_p)$, les $\mathrm{P_r}(\Delta_A^i(v))$ appartiennent à $\mathcal{S}_A$. D'autre part, l'application

$$\mathrm{T_a} : \mathbb{K}^n \to \mathbb{K}^n,\ v \mapsto \mathrm{T_a}(v) := \sum_{k=0}^{p-1} \frac{x^k}{k!}\mathrm{P_r}(\Delta_A^k(v))$$

induit l'identité sur $\ker(A_p)$ et donc tout élément de $\ker(A_p)$ est une combinaison $\mathbb{K}$-linéaire d'éléments de $\mathcal{S}_A$; d'où $\ker(A_p) = \mathcal{S}_A \otimes_{\mathcal{C}} \mathbb{K}$. □

Remarque 5. *Le lemme 3 est un corollaire immédiat de ce lemme.*

Si 0 n'est pas une singularité de $[A]$, alors, comme cela est déjà remarqué dans [Ch01] ou [Ka70], la preuve donnée ici exhibe une formule explicite pour le calcul d'une matrice fondamentale de solutions rationnelles lorsque la p-courbure est nulle ; si $A_p = 0$, alors $Y_0 = \sum_{i=0}^{p-1} \frac{(-x)^i}{i!}\,A_i$ est une matrice fondamentale de solutions rationnelles de $[A]$. En effet, on vérifie aisément que Y_0 est solution de $Y' = AY$. Ensuite, si on remplace x par 0 dans Y_0 (ce qui est possible puisque 0 n'est pas une singularité de $[A]$), alors on obtient $Y_0 = I_n$ ce qui montre que Y_0 est inversible. En revanche, lorsque 0 est une singularité de $[A]$, la matrice Y_0 peut être non-inversible.

Exemple 6. *Prenons $\mathbb{K} := \mathbb{F}_p(x)$ avec $p = 3$ et considérons le système de dimension $n = 2$ donné par la matrice*

$$A := \begin{bmatrix} \dfrac{1}{x^2} & -\dfrac{1}{x^3} \\ \dfrac{1}{x} & \dfrac{x-1}{x^2} \end{bmatrix}.$$

Ce système possède une p-courbure nulle mais, puisque 0 est une singularité de $[A]$, la «formule de Katz» ne nous fournit pas une matrice fondamentale de solutions rationnelles : en effet en appliquant cette formule, nous obtenons la matrice non-inversible suivante :

```
> LiePol(A,2,p):
> S:=evalm(add((-x)^i/(i!)*_Lie_Pol[i]/_d^i,i=0..p-1));
```

$$S := \begin{bmatrix} \dfrac{x+2}{x} & 0 \\ 2 & 0 \end{bmatrix}$$

Toujours avec $\mathbb{K} := \mathbb{F}_p(x)$ pour $p = 3$, considérons maintenant le système $[A]$ de dimension $n = 2$ donné par

$$A := \begin{bmatrix} \dfrac{x^2}{(x+2)(x-2)} & -\dfrac{1}{(x+2)(x-2)} \\ x(x+2) & -1 \end{bmatrix}.$$

Ce système possède une p-courbure nulle et 0 *est un point ordinaire. La « formule de Katz » nous donne alors la matrice fondamentale de solutions rationnelles suivante :*

```
> LiePol(A,2,p):
> S:=evalm(add((-x)^i/(i!)*_Lie_Pol[i]/_d^i,i=0..p-1));
```

$$S := \begin{bmatrix} \dfrac{x^2+2x+2}{x^3+x^2+2x+2} & \dfrac{x}{x^2+2x+1} \\ \dfrac{x^2}{x+1} & \dfrac{x^2+1}{x+1} \end{bmatrix}$$

D'une manière générale, lorsque $A_p = 0$, si l'on peut trouver $a \in \mathbb{F}_p$ tel que a n'est pas une singularité de $[A]$, alors $Y_a := \sum_{i=0}^{p-1} \frac{(-(x-a))^i}{i!} A_i(x-a)$ est une matrice fondamentale de solutions rationnelles de $[A]$. Par conséquent, cette « formule de Katz », nous permet d'exhiber une matrice fondamentale de solutions rationnelles de $[A]$ lorsque $A_p = 0$ à condition qu'il existe un élément de $\mathbb{F}_p$ qui ne soit pas une singularité de $[A]$, autrement dit, à condition que $x(x-1)\cdots(x-p+1) = x^p - x$ ne divise pas le dénominateur de A (c'est-à-dire le ppcm de tous les dénominateurs de ses coefficients).

Le calcul d'une matrice fondamentale de solutions rationnelles par cette formule nécessite uniquement la connaissance des éléments A_i (pour $i = 0, \ldots, p-1$) de la suite de Lie de $[A]$ et par conséquent la complexité du calcul des solutions rationnelles par cette formule est équivalente à celle du calcul de la p-courbure. Pour savoir si dans ce cas, utiliser cette formule est plus efficace que d'utiliser la méthode de la section 1.2 pour le calcul des solutions rationnelles, nous devons donc comparer la complexité du calcul de la p-courbure, *i. e.*, $\mathcal{O}(n^\omega\, p^2\, a\, \log(a))$ opérations dans k, à celle du calcul des solutions rationnelles, *i. e.*, $\mathcal{O}(n^\omega\, p^{\omega-2}\, \max(a,p)^2)$ opérations dans k (rappel : a est une borne sur le degré en x des entrées de la matrice A). On voit donc que dans tous les cas ($p >> a, n$, $a >> n, p$ et $n >> a, p$), le calcul de la p-courbure est plus rapide que celui des solutions rationnelles. Par conséquent, dans le cas où la p-courbure est nulle, le calcul des solutions rationnelles via « la formule de Katz » est plus efficace.

Nous donnons maintenant une première procédure appelée Scal$p$Curv (Scalar p-Curvature) qui diagonalise les systèmes possédant une p-courbure scalaire. Nous rappelons que dans tous nos algorithmes $\mathbb{K} = k(x)$ avec $k = \mathbb{F}_q$ où $q = p^r$.

Procédure Scal$p$Curv
Entrée : $A \in \mathbb{M}_n(\mathbb{K})$ satisfaisant $A_p = -\lambda\, I_n$ et $p > n$.
Sortie : $[f\, I_n] = P[A]$ avec $f \in \mathbb{K}$.
1- Calculer $\mu := -\frac{\mathrm{tr}(A)}{n}$.
2- Soit $B := A + \mu\, I_n$.
3- Soit P une matrice fondamentale de solutions rationnelles de B.
4- Retourner $P[A] = [-\mu\, I_n]$.

La correction de l'algorithme Scal$p$Curv découle directement des lemmes 8 et 9 et du théorème 1.

Dans l'étape 3, s'il existe $a \in \mathbb{F}_p$ tel que a n'est pas une singularité de $[A]$, alors nous pouvons utiliser la « formule de Katz » et nous choisissons donc $P = Y_a := \sum_{i=0}^{p-1} \frac{(-(x-a))^i}{i!}\, B_i(x-a)$. Dans ce cas, nous pouvons utiliser le lemme 5 pour calculer les B_i, ce qui est d'autant plus intéressant que l'on connait déjà les A_i puisque l'on a calculé A_p. Dans le cas contraire où $x^p - x$ divise le dénominateur de A, nous calculons une matrice fondamentale P de solutions rationnelles de $[B]$ en utilisant l'algorithme de la section 2.

Exemple 7. *Prenons* $\mathbb{K} := \mathbb{F}_p(x)$ *avec* $p = 5$ *et considérons le système différentiel de dimension* $n = 2$ *donné par la matrice suivante :*

$$A := \begin{bmatrix} 0 & 1 \\ \dfrac{2\,x+1}{(x+1)^4} & -\dfrac{2}{(x+1)^2} \end{bmatrix}.$$

Calculons sa p-courbure à l'aide de la procédure LiePol.

```
> pcurv := LiePol(A,n,p);
```

$$Pcurv = \begin{bmatrix} \dfrac{1}{1+2\,x^5+x^{10}} & 0 \\ 0 & \dfrac{1}{1+2\,x^5+x^{10}} \end{bmatrix}$$

On voit donc que sa p-courbure est scalaire ; elle s'écrit $-\lambda\, I_n$ *avec* $\lambda = -1/(1 + 2\,x^5 + x^{10})$. *En utilisant l'algorithme précédent, on doit donc trouver que le système est équivalent à* $[-\mu\, I_n]$ *avec* μ *solution de* $\tau(y) = \lambda$. *Ceci se vérifie en appliquant la procédure* ScalpCurv.

```
> evalm(ScalpCurv(A,n,p));
```

$$\begin{bmatrix} \dfrac{4}{x^2+2\,x+1} & 0 \\ 0 & \dfrac{4}{x^2+2\,x+1} \end{bmatrix}$$

Notons que $-\frac{\mathrm{tr}(A)}{n} = \frac{1}{(x^2+2\,x+1)}$ *ce qui est bien cohérent avec la théorie.*

1.4 L'eigenring d'un système différentiel linéaire

Dans cette section, nous rappelons la définition de l'eigenring associé à un système différentiel ainsi qu'une méthode pour le calculer en pratique. Ensuite, nous prouvons les liens entre l'eigenring et la p-courbure d'un système différentiel linéaire et nous donnons quelques résultats sur sa dimension.

1.4.1 Définition et calcul

Tout comme en caractéristique zéro (voir par exemple [Si96, Ba01]), en caractéristique p, l'eigenring associé à un système différentiel peut se définir de la façon suivante :

Définition 13. *Soit* $[A]$ *un système différentiel avec* $A \in \mathbb{M}_n(\mathbb{K})$. L'eigenring $\mathcal{E}(A)$ de $[A]$ *est l'ensemble défini par*

$$\mathcal{E}(A) \;=\; \{P \in \mathbb{M}_n(\mathbb{K}) \mid P' = A\,P - P\,A\}.$$

L'eigenring est une $\mathcal{C}$-algèbre de dimension finie inférieure ou égale à n^2. En particulier, il contient la matrice identité I_n. Par conséquent, le polynôme minimal (et aussi caractéristique) d'un élément de l'eigenring aura ses coefficients dans le corps des constantes $\mathcal{C}$.

D'après la définition 13, calculer l'eigenring revient à trouver les solutions rationnelles du système $P' = A\,P - P\,A$. Ceci peut se faire de la manière suivante : considérons l'application

$$\mathrm{Vect} : \mathbb{M}_n(\mathbb{K}) \to \mathbb{K}^{n^2},\ P = \begin{pmatrix} \cdots & L_1 & \cdots \\ & \vdots & \\ \cdots & L_n & \cdots \end{pmatrix} \mapsto \begin{pmatrix} {}^tL_1 \\ \vdots \\ {}^tL_n \end{pmatrix},$$

où les L_i désignent les lignes de la matrice P. On montre alors que

$$P' = A\,P - P\,A \Longleftrightarrow \mathrm{Vect}(P)' = (A \otimes I_n - I_n \otimes {}^tA)\,\mathrm{Vect}(P),$$

et, par conséquent, trouver une base de l'eigenring équivaut à calculer une base de solutions rationnelles du système différentiel $Z' = (A \otimes I_n - I_n \otimes {}^tA)\,Z$ de dimension n^2. L'approche décrite ci-dessus ne dépend pas de la caractéristique. Cependant, compte tenu du fait qu'il est (théoriquement) plus facile de calculer les solutions rationnelles en caractéristique p qu'en caractéristique zéro, calculer l'eigenring est aussi plus facile ; ceci revient à résoudre un système linéaire homogène de dimension $n^2\,p$ (Voir la section 1.2) de telle sorte que d'après la proposition 2, nous pouvons affirmer :

Proposition 4. *Soit* $[A]$ *un système différentiel avec* $A \in \mathbb{M}_n(k(x))$ *où* $k = \mathbb{F}_p$ *ou* $\overline{\mathbb{F}}_p$. *Soit* D *le plus petit dénominateur commun des coefficients de* $\mathcal{A} = A \otimes I_n - I_n \otimes {}^tA$ *et soit* $\tilde{\mathcal{A}} := D\,\mathcal{A} \in \mathbb{M}_n(k[x])$. *Soit* a *une borne sur le degré des entrées de* $\tilde{\mathcal{A}}$. *Alors l'eigenring de* $[A]$ *peut se calculer en* $\mathcal{O}(n^6\,\max(a,p)^2\,p)$ *opérations dans* k.

Dans [GZ03], Giesbrecht et Zhang utilisent aussi cette méthode pour calculer l'eigenring d'un polynôme de Ore en caractéristique p.

Un algorithme plus efficace pour calculer l'eigenring d'un système différentiel à coefficients dans $\mathcal{F}(x)$ où $\mathcal{F}$ est un corps de caractéristique zéro se trouve dans [BP98]. Pour le cas des équations scalaires, nous renvoyons à [Ho96].

Nous rappelons maintenant quelques résultats sur la dimension de l'eigenring en tant qu'espace vectoriel sur $\mathcal{C}$. Nous commençons par un lemme très utile :

Lemme 10 ([Ba01], Proposition 2). *Si $[A]$ et $[B]$ sont deux systèmes équivalents, alors leurs eigenrings sont isomorphes (en tant que $\mathcal{C}$-algèbres). Plus précisément, si $[B] = P[A]$, alors l'isomorphisme entre $\mathcal{E}(A)$ et $\mathcal{E}(B)$ est donné par $T \mapsto P^{-1}\,T\,P$.*

Si $[A]$ est irréductible, alors $\mathcal{E}(A)$ est un corps (voir par exemple [Ba01, Theorem 1]). La réciproque est fausse mais on peut quand même montrer (voir [Ba01]) que :

Lemme 11. *Soit $[A]$ un système différentiel avec $A \in \mathbb{M}_n(\mathbb{K})$. Si $[A]$ est équivalent à $[B] = \bigoplus_{i=1}^{s}[A^{[i]}]$, alors* $\dim(\mathcal{E}(A)) \geq s$.

Démonstration. L'eigenring de $[B] = \bigoplus_{i=1}^{s}[A^{[i]}]$ contient toutes les matrices de la forme

$$\begin{pmatrix} c_1\,I_{n_1} & & 0 \\ & \ddots & \\ 0 & & c_s\,I_{n_s} \end{pmatrix}$$

avec $c_1, \ldots, c_s \in \mathcal{C}$ (les n_i désignent les dimensions des blocs $A^{[i]}$). Donc, $\dim(\mathcal{E}(B)) = \dim(\mathcal{E}(A)) \geq s$. □

1.4.2 Eigenring et p-courbure

L'eigenring d'un système différentiel $[A]$ peut aussi être vu comme l'ensemble des applications $\mathbb{K}$-linéaires de $\mathbb{K}^n$ dans $\mathbb{K}^n$ qui commutent avec Δ_A. On en déduit facilement que :

Lemme 12. *Soit $[A]$ un système différentiel avec $A \in \mathbb{M}_n(\mathbb{K})$. La p-courbure A_p de $[A]$ appartient à l'eigenring,*

$$A_p \in \mathcal{E}(A).$$

Plus précisément, si l'on note $\mathrm{Com}(A_p)$ *l'ensemble des matrices dans $\mathbb{M}_n(\mathbb{K})$ qui commutent avec A_p et $\mathcal{C}[A_p]$ l'ensemble des polynômes en A_p à coefficients dans $\mathcal{C}$, alors on a les inclusions (d'ensembles) suivantes :*

$$\mathcal{C}[A_p] \subseteq \mathcal{E}(A) \subseteq \mathrm{Com}(A_p). \tag{1.6}$$

Remarque 6. *Une conséquence intéressante de ce lemme est que, contrairement à ce qui se passe en caractéristique zéro, nous pouvons ici calculer un élément de l'eigenring, à savoir la p-courbure, sans déterminer entièrement l'eigenring.*
Le lemme 12 donne aussi une autre méthode pour calculer l'eigenring ; calculer l'ensemble $\mathrm{Com}(A_p)$ *et pour* $T \in \mathrm{Com}(A_p)$ *tester si* $T' = A\,T - T\,A$.

Nous allons maintenant affiner le lemme précédent en prouvant le résultat suivant qui est une conséquence des lemmes 5 et 9.

Proposition 5. $\mathrm{Com}(A_p) = \mathcal{E}(A) \otimes_{\mathcal{C}} \mathbb{K}$ *et en particulier,* $\dim_{\mathcal{C}}(\mathcal{E}(A)) = \dim_{\mathbb{K}}(\mathrm{Com}(A_p))$.

Démonstration. Considérons à nouveau le système différentiel $Z' = (A \otimes I_n - I_n \otimes {}^tA)\,Z := \mathcal{A}\,Z$. D'après le lemme 9, nous savons que $\ker(\mathcal{A}_p) = S_{\mathcal{A}} \otimes_{\mathcal{C}} \mathbb{K}$ où $S_{\mathcal{A}}$ représente l'espace des solutions rationnelles de $[\mathcal{A}]$. De plus, nous avons $\mathcal{E}(A) \cong S_{\mathcal{A}}$ et $\mathrm{Com}(A_p) \cong \ker(A_p \otimes I_n - I_n \otimes {}^tA_p)$ donc il nous reste juste à vérifier que $\mathcal{A}_p = A_p \otimes I_n - I_n \otimes {}^tA_p$. Ceci provient directement du fait que prendre la p-courbure est un foncteur commutant avec les constructions sur les modules différentiels (voir la sous-section 1.3.2) mais nous en donnons ici une preuve élémentaire. Avec la notation standard A^* pour la matrice adjointe $-{}^tA_p$ de A_p, on applique le lemme 5 à $A \otimes I_n$ et $I_n \otimes A^*$. Une vérification immédiate montre que ces matrices satisfont les hypothèses du lemme : d'où $\mathcal{A}_p = (A \otimes I_n)_p + (I_n \otimes A^*)_p$. Il ne reste alors plus qu'à remarquer que $\forall i \in \mathbb{N}, (A \otimes I_n)_i = A_i \otimes I_n$ pour prouver que $\mathcal{A}_p = A_p \otimes I_n + I_n \otimes (A^*)_p$. On conclut en utilisant la relation $(A^*)_p = -{}^tA_p$. □

Corollaire 1. *Soit* $\chi(A_p) := \det(\lambda\,I - A_p)$ *(resp.* $\chi_{\min}(A_p)$*) le polynôme caractéristique (resp. minimal) de la p-courbure. Si* $\chi(A_p) = \chi_{\min}(A_p)$*, alors* $\mathcal{E}(A) = \mathcal{C}[A_p]$.

Démonstration. D'après la proposition précédente, on a $\mathrm{Com}(A_p) = \mathcal{E}(A) \otimes_{\mathcal{C}} \mathbb{K}$. De plus, on a clairement $\mathbb{K}[A_p] = \mathcal{C}[A_p] \otimes_{\mathcal{C}} \mathbb{K}$. Par conséquent, si $\mathbb{K}[A_p] = \mathrm{Com}(A_p)$, alors $\mathcal{E}(A) = \mathcal{C}[A_p]$ (puisque $\mathcal{C}[A_p] \subset \mathcal{E}(A)$). On termine la preuve en utilisant que (voir [Ja85, Ex. 7 p. 202 et Corollary of Theorem 3.16 p. 207]), $\mathbb{K}[A_p] = \mathrm{Com}(A_p)$ est équivalent à $\chi(A_p) = \chi_{\min}(A_p)$. □

1.5 Décomposition maximale

Dans tout ce qui suit, nous noterons respectivement $\chi(M) := \det(\lambda\,I - M)$ et $\chi_{\min}(M)$ les polynômes caractéristique et minimal d'une matrice M.

Le but dans cette partie est de calculer une *décomposition maximale de* $[A]$, c'est-à-dire, trouver P tel que $[B] = P[A] = \bigoplus_{i=1}^{r} [B^{[i]}]$ où les $[B^{[i]}]$ sont indécomposables. Comme cela est établi dans [Pu95] (voir aussi [PS03, 13]), la forme de Jordan de la p-courbure donne la factorisation complète du module différentiel associé à $[A]$. Ceci peut être rendu algorithmique en deux étapes : tout d'abord, on calcule une décomposition isotypique (voir Définition 14 ci-dessous). Puis, dans un second temps, on affine cette décomposition pour obtenir une décomposition maximale. Compte tenu du fait que la méthode donnée par van der Put pour effectuer cette seconde étape peut nécessiter des

calculs (possiblement coûteux) dans des extensions algébriques, nous proposons ici une méthode alternative basée sur l'eigenring.

1.5.1 Décomposition isotypique

Définition 14. *Soit $[A]$ un système différentiel avec $A \in \mathbb{M}_n(\mathbb{K})$. Une décomposition isotypique de $[A]$ est une décomposition de la forme $[A] = \bigoplus_{i=1}^{r}[A^{[i]}]$ où, pour tout $i \in \{1,\dots,r\}$, $\chi((A^{[i]})_p) = F_i^{m_i}$ avec F_i irréductible sur $\mathcal{C}[X]$, $F_i \neq F_j$ pour $i \neq j$ et $m_i \geq 1$.*

Soit $[A]$ un système différentiel avec $A \in \mathbb{M}_n(\mathbb{K})$. Nous donnons ici un algorithme pour calculer une décomposition isotypique de $[A]$. Cet algorithme s'inspire à la fois de l'algorithme de van der Put dans le cas scalaire (algorithme découlant directement de la classification-voir [Pu97]) et de la méthode de Barkatou dans [Ba01] pour factoriser les systèmes différentiels linéaires en caractéristique zéro.

Proposition 6 ([Ba01], Theorem 2). *Soit $[A]$ un système différentiel avec $A \in \mathbb{M}_n(\mathbb{K})$. Si, pour un certain $T \in \mathcal{E}(A)$, $\chi(T) = F_1 \cdots F_r$ avec les $F_i \in \mathcal{C}[X]$ et pour tout $i \neq j$, $\gcd(F_i, F_j) = 1$, alors nous pouvons effectivement trouver $P \in \mathrm{GL}_n(\mathbb{K})$ tel que $P[A] = \bigoplus_{i=1}^{r}[A^{[i]}]$.*

Démonstration. Pour la convenance du lecteur, nous répétons ici la preuve de [Ba01, Theorem 2]. Supposons $r = 2$ ce qui n'enlève rien à la généralité de cette preuve.
Soit $T \in \mathcal{E}(A)$ et $P \in \mathrm{GL}_n(\mathbb{K})$ tels que $P^{-1}\,T\,P = \bigoplus_{i=1}^{2} T^{[i]}$ avec $\chi(T^{[i]}) = F_i$ et $\gcd(F_1, F_2) = 1$. Considérons $B := P[A] = \begin{pmatrix} B^{[11]} & B^{[12]} \\ B^{[21]} & B^{[22]} \end{pmatrix}$ où les $B^{[ii]}$ ont les mêmes dimensions que les $T^{[i]}$. Nous devons montrer que $B^{[21]} = B^{[12]} = 0$. D'après le lemme 10, $P^{-1}\,T\,P \in \mathcal{E}(B)$ et donc

$$\begin{pmatrix} T^{[1]'} & 0 \\ 0 & T^{[2]'} \end{pmatrix} = \begin{pmatrix} B^{[11]} & B^{[12]} \\ B^{[21]} & B^{[22]} \end{pmatrix} \begin{pmatrix} T^{[1]} & 0 \\ 0 & T^{[2]} \end{pmatrix} - \begin{pmatrix} T^{[1]} & 0 \\ 0 & T^{[2]} \end{pmatrix} \begin{pmatrix} B^{[11]} & B^{[12]} \\ B^{[21]} & B^{[22]} \end{pmatrix}.$$

Ceci implique en particulier $0 = B^{[21]}\,T^{[1]} - T^{[2]}\,B^{[21]}$. Maintenant, avec l'hypothèse $\gcd(F_1, F_2) = 1$, un résultat classique d'algèbre linéaire (voir Corollary S2.3 dans [GLR82]) affirme que la seule solution du système $0 = X\,T^{[1]} - T^{[2]}\,X$ est la solution nulle et donc $B^{[21]} = 0$. De la même manière, on montre que $B^{[12]} = 0$ ce qui termine la preuve. □

Théorème 2. *Soit $[A]$ un système différentiel avec $A \in \mathbb{M}_n(\mathbb{K})$. L'algorithme suivant calcule une décomposition isotypique de $[A]$:*

Procédure IsoDec *(Isotypical Decomposition)*
Entrée *:* $A \in \mathbb{M}_n(\mathbb{K})$ *et sa p-courbure* A_p*.*
Sortie *:* $[B] = P[A]$ *où* $[B] = \bigoplus_{i=1}^{r}[B^{[i]}]$ *et* $\chi((B^{[i]})_p) = F_i^{m_i}$ *avec* F_i *irréductible et* $\gcd(F_i, F_j) = 1$*.*
1- *Calculer et factoriser (sur* $\mathcal{C}$*)* $\chi(A_p)$*.*
(Posons $\chi(A_p) = F_1^{m_1} \cdots F_r^{m_r}$ *avec* $\gcd(F_i, F_j) = 1$ *et les* F_i *irréductibles)*
2- *Si* $r = 1$ *et* $m_1 = 1$*, alors retourner* $[A]$*.*
3- *Pour* $i = 1, \ldots, r$*, calculer une base* $e_{i,1}, \ldots, e_{i,n_i}$ *de* $\ker(F_i^{m_i}(A_p))$*.*
4- *Construire la matrice* $P \in \mathbb{M}_n(\mathbb{K})$ *ayant les* $e_{i,j}$ *pour colonnes.*
5- *Retourner* $P[A] = \bigoplus_{i=1}^{r}[A^{[i]}]$*.*

Démonstration. L'étape 2 découle directement de la classification des modules différentiels en caractéristique p. En effet, si $\chi(A_p)$ est irréductible, la forme de Jordan de A_p sur $\mathbb{K}$ ne possède qu'un seul bloc diagonal qui n'est par ailleurs pas triangularisable par blocs de telle sorte que $[A]$ est irréductible. Dans le cas décomposable, notre méthode pour trouver les facteurs de manière effective découle directement de la proposition 6 appliquée à l'élément A_p de $\mathcal{E}(A)$. □

Exemple 8. *Prenons* $\mathbb{K} := \mathbb{F}_p(x)$ *pour* $p = 3$ *et considérons le système différentiel* $Y' = AY$ *de dimension* $n = 4$ *donné par*

$$A := \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},$$

où

$$A_{11} := \begin{bmatrix} \frac{(x+1)\,x}{(x+2)^2} & \frac{2\,x^4+2\,x+1}{(x+2)^2} \\ \frac{(x^3+x^2+x+2)\,(x+1)}{(x+2)^2\,x} & \frac{2\,x^5+2\,x^6+2\,x^3+1}{(x+2)^2\,x} \end{bmatrix},$$

$$A_{12} := \begin{bmatrix} 2\,\frac{2\,x^4+2\,x+1}{(x+2)^2} & 2\,\frac{x^4+x^3+x^2+2\,x+2}{(x+2)^2} \\ \frac{x^4+x^5+x^2+x+1}{(x+2)^2} & 2\,\frac{2\,x^4+2\,x^3+x^5+2\,x+1}{(x+2)^2} \end{bmatrix},$$

$$A_{21} := \begin{bmatrix} 2\,\frac{(x+1)^3\,(x^2+2\,x+2)}{(x+2)^2\,x} & 2\,\frac{(x+1)\,x^2\,(x^2+2\,x+2)}{(x+2)^2} \\ 2\,\frac{x\,(x^3+2\,x+1)}{(x+2)^2} & \frac{2\,x^3+2\,x^2+2\,x^4+x+1}{(x+2)^2} \end{bmatrix},$$

$$A_{22} := \begin{bmatrix} \frac{(x^3+2\,x+1)^2}{(x+2)^2\,x} & \frac{x^4+2\,x^6+2\,x^3+x+2}{(x+2)^2\,x} \\ 2\,\frac{2\,x^3+2\,x^2+2\,x^4+x+1}{(x+2)^2} & \frac{x\,(x^2+1)}{(x+2)^2} \end{bmatrix}.$$

Calculons tout d'abord la p-courbure A_p avec la procédure LiePol.

```
> A_p := evalm(LiePol(A,n,p));
```

$$A_p := (x+1)^3 (x+2)^2 \begin{bmatrix} 2 & 2x & x & 2x \\ 2(x+1) & 2(x+1)x & (x+1)x & 2(x+1)x \\ x+1 & 2x^2 & x^2 & 2(x+1)^2 \\ 2x & 2x & x & 2 \end{bmatrix}$$

Ici le polynôme caractéristique de A_p est

$$(X + 2x^6 + 1)^2 X^2,$$

et par conséquent, notre procédure IsoDec *nous renvoie une décomposition en deux facteurs de dimension deux.*

```
> evalm(IsoDec(A,A_p,p));
```

$$\begin{bmatrix} \frac{x^4+2x^2+2}{(x+2)^2} & 2\frac{(x^3+x^2+x+2)x}{(x+2)^2} & 0 & 0 \\ \frac{x^2+1}{(x+2)^2} & \frac{(x^3+x^2+2x+1)(x+1)}{(x+2)^2} & 0 & 0 \\ 0 & 0 & \frac{1}{x} & \frac{1}{x+2} \\ 0 & 0 & 0 & \frac{1}{(x+2)x} \end{bmatrix}$$

Notons que les facteurs calculés sont tous les deux décomposables (voir Exemple 9) de telle sorte que cette décomposition de $[A]$ n'est pas maximale.

1.5.2 Le cas $\chi(A_p) = F^m$

Nous avons ramené le problème au cas où le système $[A]$ possède une p-courbure ayant un polynôme caractéristique de la forme F^m avec F irréductible et $m \geq 2$. Nous savons de plus que dans ce cas, $\chi_{\min}(A_p) = F^d$ avec $1 \leq d \leq m$.
Ici, le fait que la forme de Jordan de la p-courbure donne une factorisation complète du système peut être interprété comme suit : tout d'abord, nous avons le critère d'indécomposabilité suivant :

Proposition 7. *Soit $[A]$ un système différentiel avec $A \in \mathbb{M}_n(\mathbb{K})$ satisfaisant $\chi(A_p) = F^m$ avec F irréductible et $m \geq 2$. Alors, $[A]$ est indécomposable si, et seulement si, $\chi_{\min}(A_p) = \chi(A_p)$.*

Démonstration. D'après la classification des modules différentiels en caractéristique p, $[A]$ est indécomposable si, et seulement si la forme de Jordan de A_p possède un seul

bloc indécomposable ce qui est équivalent au fait que A_p ne possède qu'un seul diviseur élémentaire (voir [Ja85, 3.10] ou [Ga66, section 7]) sur $\mathbb{K}$. Ceci est aussi équivalent à $\chi_{\min}(A_p) = \chi(A_p)$. □

En fait, comme chaque diviseur élémentaire de A_p sur $\mathbb{K}$ correspond à un bloc indécomposable dans sa forme de Jordan rationnelle sur $\mathbb{K}$ (voir [Ga66, section 7]), la classification de van der Put nous permet d'affirmer que le nombre de facteurs indécomposables de $[A]$ est égal au nombre de diviseurs élémentaires de A_p. Voici brièvement comment ceci est rendu algorithmique dans [Pu97] : soit a l'image de X dans $\mathcal{C}^+ := \mathcal{C}[X]/F$ et f le degré du polynôme F (on a alors $n = f\,m$). Nous avons $F^m = (X-a)^m\,Q_a^m$ de sorte que l'algorithme IsoDec appliqué sur $\mathbb{K}^+ := \mathbb{K}[X]/F$ nous donne une décomposition $[A] = [A^+] \oplus [A^{[2]}]$ où $[A^+]$ est défini sur $\mathbb{K}^+$, possède une dimension égale à m et satisfait $\chi((A^+)_p) = (X-a)^m$. Soit maintenant $b \in K^+$ une solution de $\tau(b) = a$ (b peut être choisi dans $\mathbb{K}^+$ d'après le théorème 1). En « décalant » le système, *i. e.*, en considérant $\tilde{A^+} = A^+ + b\,I_m$ (comme dans la preuve du lemme 8), nous obtenons alors un système de même dimension que $[A^+]$ mais satisfaisant $\chi((\tilde{A^+})_p) = X^m$ et $\chi_{\min}((\tilde{A^+})_p) = X^d$. Nous calculons alors le noyau de $\Delta_{\tilde{A^+}}$ dans $\mathbb{K}^+ + \mathbb{K}^+\,\ell + \cdots + \mathbb{K}^+\,\ell^{d-1}$ où ℓ est défini par $\ell' := 1/x$. Nous savons que celui-ci possède une dimension égale à m. Nous en déduisons donc une base du module différentiel associé à $[\tilde{A^+}]$ dans laquelle le système a une forme triangulaire par blocs maximale (pour construire la bonne base, on peut calculer les diviseurs élémentaires de $(\tilde{A^+})_p$). Finalement, on applique le « décalage inverse » et on transforme la base du module différentiel de dimension m sur $\mathbb{K}^+$ en une base du module différentiel de dimension $n = m\,f$ sur $\mathbb{K}$: le système écrit dans cette base constitue une décomposition maximale de $[A]$.

Exemple 9. *Continuons la factorisation de l'exemple 8. Nous avons calculé une première décomposition du système en deux blocs diagonaux de dimension* 2. *Ici, le polynôme minimal de la p-courbure du premier bloc* $[A^{[1]}]$ *est* $X+2\,x^6+1$ *et celui du second bloc* $[A^{[2]}]$ *est* X. *Ceci implique que les diviseurs élémentaires de* $A_p^{[1]}$ *sont* $X+2\,x^6+1$ *et* $X+2\,x^6+1$ *et ceux de* $A_p^{[2]}$ *sont* X *et* X. *Par conséquent, chaque facteur est décomposable et on peut trouver* P *tel que* $P[A] = [x^2] \oplus [x^2] \oplus [0] \oplus [0]$. *Remarquons que* $-x^2 = -tr(A^{[1]})/2$ *est solution de l'équation scalaire* $\tau(y) = 2\,x^6 + 1$ *(voir Théorème 1).*

Si $\deg(F) > 1$, l'image a de X dans $\mathcal{C}^+ := \mathcal{C}[X]/F$ n'appartient pas à $\mathcal{C}$ et nous devons travailler dans une extension algébrique de $\mathbb{K}$. Ceci motive notre démarche proposant une approche différente et rationnelle pour le calcul de la décomposition maximale lorsque $\deg(F) > 1$: cette méthode suit ce qui est fait dans [Ba01] et est basée sur l'utilisation de l'eigenring tout entier.

L'algorithme IsoDec utilise la proposition 6 avec $T = A_p$ et calcule une décomposition isotypique du système. Lorsque cette décomposition n'est pas maximale, nous pouvons encore utiliser cette même proposition 6 mais en prenant pour T un autre élément de l'eigenring. La proposition suivante montre que l'eigenring contient un élément « de spectre maximal » c'est-à-dire un élément $T \in \mathcal{E}(A)$ tel que :

– $\chi(T) = F_1 \cdots F_r$ avec les $F_i \in \mathcal{C}[X]$ irréductibles, $\gcd(F_i, F_j) = 1$ pour $i \neq j$ et,

– il n'existe pas $U \in \mathcal{E}(A)$ tel que $\chi(U) = G_1 \cdots G_s$ avec $s > r$, les $G_i \in \mathcal{C}[X]$ irréductibles et $\gcd(G_i, G_j) = 1$ pour $i \neq j$.

Par conséquent, si nous utilisons un tel élément dans la proposition 6, cela nous donne directement une décomposition maximale de $[A]$.

Proposition 8. *Soit $[A]$ un système différentiel avec $A \in \mathbb{M}_n(\mathbb{K})$. Supposons que $[A]$ soit équivalent à $\bigoplus_{i=1}^{s}[A^{[i]}]$, alors il existe $T \in \mathcal{E}(A)$ tel que $\chi(T) = F_1 \cdots F_s$ avec les F_i dans $\mathcal{C}[X]$ et $\gcd(F_i, F_j) = 1$ pour $i \neq j$.*

Démonstration. On voit facilement qu'étant donné $c_1, \ldots, c_s$ avec $c_i \in \mathcal{C}$ et $c_i \neq c_j$, l'eigenring de $[A]$ contient un élément T semblable à

$$\begin{pmatrix} c_1\, I_{n_1} & & 0 \\ & \ddots & \\ 0 & & c_s\, I_{n_s} \end{pmatrix}.$$

□

En pratique, un élément « de spectre maximal » se trouve souvent en prenant un élément « au hasard » dans l'eigenring. C'est ainsi que la plupart des algorithmes pour calculer des décompositions (en ppcm) en caractéristique zéro procèdent. Cependant, cette approche est seulement une heuristique et si elle échoue on peut utiliser la proposition 9 ci-dessous (comme cela est proposé dans [GZ03]) ou la méthode de van der Put (exposée après la proposition 7).

Exemple 10. *Soit $\mathbb{K} := \mathbb{F}_p(x)$ avec $p = 5$ et considérons le système $Y' = AY$ où la matrice A à coefficients dans $\mathbb{K}$ est donnée par*

$$A := \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\dfrac{3}{4\,x^2} & \dfrac{1}{x} \end{bmatrix}.$$

Calculons tout d'abord sa p-courbure et son polynôme caractéristique.

```
> A_p:=evalm(LiePol(A,n,p));
```

$$A_p := \begin{bmatrix} 0 & 0 & \dfrac{3}{x^3} & \dfrac{2}{x^2} \\ 0 & 0 & \dfrac{2}{x^4} & \dfrac{1}{x^3} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> F:=Factors(numer(charpoly(A_p,X) mod p)) mod p;
```

$$F := [1,\ [[X,\ 4]]]$$

En regardant d'un peu plus près, on voit que le polynôme minimal de A_p est égal à X^2 de telle sorte que ce système peut se décomposer en 2 ou 3 blocs indécomposables selon que les diviseurs élémentaires de A_p sont X^2, X^2 ou X^2, X, X.

En suivant notre algorithme, nous calculons donc l'eigenring de $[A]$.

```
> E:=[eigmodp(A,n,p)]:
```

Prenons maintenant un élément « au hasard » dans cet eigenring et regardons si son polynôme caractéristique se factorise en 2 ou 3 facteurs irréductibles. Notons qu'ici, pour choisir cet élément, nous prenons au hasard une combinaison linéaire à coefficients dans $\mathbb{F}_p$ des éléments de la base de l'eigenring calculée précédemment;

```
> die := rand(1..p):
> elmt:=subs(c=x^p,evalm(add(die()*E[i],i=1..nops(E)))):
> F:=Factors(charpoly(elmt,T) mod p) mod p;
```

$$F := [1,\ [[1,\ 1],\ [T + 3\,x^5 + 3,\ 2],\ [T + x^5 + 3,\ 2]]]$$

On voit donc que cet élément va décomposer notre système en deux blocs (voir Proposition 6) : calculons donc cette décomposition.

```
> E1:=Nullspace(evalm(subs(T=elmt,(T+3*x^5+3)^2))) mod p:
> E2:=Nullspace(evalm(subs(T=elmt,(T+x^5+3)^2))) mod p:
> P1:=concat(E1[1],E1[2],E2[1],E2[2]): IP1:=Inverse(P1) mod p:
> P1[A]:=evalm(IP1 &* (A &* P1 - diff_mat(P1,p)));
```

$$P1[A] := \begin{bmatrix} 0 & \dfrac{3\,x^4+4}{x^2} & 0 & 0 \\ 0 & \dfrac{2}{x} & 0 & 0 \\ 0 & 0 & 0 & 4\,x \\ 0 & 0 & 0 & \dfrac{3}{x} \end{bmatrix}$$

En poussant un peu plus loin l'analyse de cet exemple, on peut voir que les deux blocs ainsi calculés sont indécomposables.

Remarque 7. *La proposition 8 n'utilise pas le fait que la décomposition isotypique a déjà été calculée de telle sorte que nous pouvons l'utiliser au début et obtenir ainsi la décomposition maximale en une seule étape. Cependant, comme la décomposition isotypique est assez facile à calculer à partir de la p-courbure, il semble préférable de commencer par la calculer : en effet, ceci a l'avantage de nous amener à calculer des*

eigenrings de systèmes de dimension plus petite. De plus, pour calculer ces eigenrings, nous pouvons tirer partie du lemme 12 comme suit : connaissant la p-courbure A_p du système dont nous devons calculer l'eigenring, nous savons que I, A_p, ..., A_p^{d-1} (avec $d = \deg_x(\chi_{\min}(A_p))$) sont des éléments $\mathcal{C}$-indépendants de l'eigenring ce qui nous permet de réduire encore les calculs (en réduisant la dimension du système dont on cherche l'eigenring) à effectuer pour obtenir une base complète de l'eigenring.

Voici une autre manière d'utiliser l'eigenring pour factoriser notre système. Le théorème suivant ([Ja80, Proposition 3.1 p. 111]) fournit un critère d'indécomposabilité.

Proposition 9. *Soit $A \in \mathbb{M}_n(\mathbb{K})$, $A \neq 0$. Alors, $[A]$ est indécomposable si, et seulement si $\mathcal{E}(A)$ ne contient pas d'idempotent autre que 0 et I_n.*

Démonstration. Supposons $[A]$ décomposable. Il existe alors $T \in \mathcal{E}(A)$ semblable à $\begin{pmatrix} I_{n_1} & 0 \\ 0 & 0 \end{pmatrix}$. Il est alors clair que $T \neq 0$, $T \neq I_n$ et $T^2 = T$.
Réciproquement, si $\mathcal{E}(A)$ contient un idempotent T différent de 0 et I, alors d'après la proposition 6, $[A]$ est décomposable. □

Par conséquent, calculer de manière récursive (*i. e.*, jusqu'à ce que les facteurs soient indécomposables) des idempotents dans l'eigenring permet aussi de calculer une décomposition maximale de $[A]$. Cette méthode est celle utilisée par Giesbrecht et Zhang dans [GZ03] pour décomposer des polynômes de Ore en caractéristique p. Pour ce faire, ils montrent comment utiliser l'algorithme de décomposition d'algèbres associatives donné dans [IRS94] pour calculer de « bons » idempotents dans l'eigenring. Leur algorithme est donc construit d'une manière assez différente du nôtre : ils raisonnent directement sur l'eigenring du système de départ alors que nous faisons le choix de décomposer d'abord au maximum en utilisant la p-courbure (plus facile à calculer). Nous ne calculons l'eigenring que lorsque nous ne pouvons plus continuer à décomposer avec la p-courbure sans aller calculer dans des extensions algébriques du corps de base.

Remarque 8. *Soit $\mathcal{F}$ un corps quelconque. Une matrice $M \in \mathbb{M}_n(\mathcal{F})$ est dite* semi-simple *si tout sous-espace invariant par M admet un sous-espace supplémentaire invariant par M. En particulier, Une matrice $M \in \mathbb{M}_n(\mathcal{F})$ est semi-simple si elle est diagonalisable sur $\overline{\mathcal{F}}$.*
On déduit alors facilement de ce qui précède le résultat suivant : si $[A]$ est un système différentiel avec $A \in \mathbb{M}_n(\mathbb{K})$, alors sa p-courbure A_p est semi-simple si, et seulement si, le système est complètement réductible sur $\mathbb{K}$.

1.6 Le cas indécomposable

Dans cette section, nous donnons une alternative rationnelle à la méthode de van der Put dans le cas où le système $[A]$ est indécomposable, *i. e.*, $\chi(A_p) = \chi_{\min}(A_p) = F^m$ avec F irréductible et $m > 1$ (rappelons que si $m = 1$, alors $[A]$ est irréductible). Nous montrons comment calculer *une réduction maximale*, c'est-à-dire, comment obtenir un système équivalent à $[A]$ ayant une forme triangulaire par blocs avec des blocs diagonaux irréductibles. Ce résultat peut s'énoncer comme suit :

Proposition 10. *Soit $[A]$ un système différentiel avec $A \in \mathbb{M}_n(\mathbb{K})$. Si $\chi(A_p) = \chi_{\min}(A_p) = F^m$ avec F irréductible et $m > 1$, alors $[A]$ est réductible et une réduction maximale peut être calculée de manière effective.*

Démonstration. Posons $v := F(A_p)$. On a $v^m = 0$ et pour tout $i \in \{1, \ldots, m-1\}$, $v^i \neq 0$. Soit $E_i := \ker(v^i)$ pour tout $i \in \{1, \ldots, m\}$. Il est clair que $E_m = \mathbb{K}^n$ et $E_i \subset E_{i+1}$ de telle sorte que $(E_i)_i$ est un drapeau de $\mathbb{K}^n$. La matrice de v dans une base de $\mathbb{K}^n$ adaptée à ce drapeau (*i. e.*, une base calculée à partir d'une base de E_1 étendu en une base de E_2, ...) est triangulaire par blocs. De plus, comme $v(E_i) \subset E_{i-1}$ les blocs diagonaux sont nuls. En d'autres termes, il existe $P \in \mathrm{GL}_n(\mathbb{K})$ telle que $P^{-1}\, v\, P = S$ où S est triangulaire par blocs avec des blocs diagonaux nuls. Maintenant, on remarque que tous les v^i sont des éléments non nuls et non inversibles de l'eigenring donc pour les mêmes raisons que dans la preuve de [Ba01, Theorem 1], un calcul direct montre que $B := P[A]$ a une forme triangulaire par blocs. De plus, les blocs diagonaux $B^{[ii]}$ de B satisfont $\chi((B^{[ii]})_p) = F$ et sont donc irréductibles. □

La proposition précédente s'applique à tout élément de l'eigenring et pas seulement à la p-courbure. Par conséquent, le système peut être réduit en trouvant des diviseurs de zéro non-triviaux dans l'eigenring (par exemple en utilisant [IRS94]) : c'est ainsi que procèdent Giesbrecht et Zhang dans [GZ03]. Notons que, si l'on veut calculer une factorisation maximale du système, alors il nous faut appliquer leur méthode de manière récursive.

Exemple 11. *Posons $\mathbb{K} := \mathbb{F}_p(x)$ avec $p = 5$ et considérons le système de dimension $n = 2$ défini par la matrice suivante :*

$$A := \begin{bmatrix} \dfrac{1+3\,x}{4\,x+x^2} & \dfrac{4}{4\,x+x^2} \\ \dfrac{2\,x+4}{4+x} & \dfrac{1}{4+x} \end{bmatrix}.$$

Calculons sa p-courbure :

```
> pcurv:=LiePol(A,n,p);
```

$$pcurv := \begin{bmatrix} \dfrac{3\,x^3+3\,x^2+4\,x+4}{x^9+4\,x^4} & \dfrac{x^2+2\,x+1}{x^9+4\,x^4} \\ \dfrac{x^4+x^2+4}{x^9+4\,x^4} & \dfrac{2\,x^3+2\,x^2+x+1}{x^9+4\,x^4} \end{bmatrix}$$

Son polynôme caractéristique est égal à X^2 tout comme son polynôme minimal et nous sommes donc dans le cas indécomposable.
En suivant la preuve précédente, nous sommes amenés à calculer le noyau de la p-courbure :

```
> E:=Nullspace(map(X-> Normal(X) mod p,evalm(subs(X=Pcurv,X)))) mod
p;
```

$$E := \{\left[\frac{4\,(2\,x+2)}{x^2+3},\ 1\right]\}$$

et à le compléter pour obtenir une base adaptée au drapeau (voir la preuve précédente). On obtient alors la matrice de passage suivante :

$$P := \begin{bmatrix} \dfrac{4\,(2\,x+2)}{x^2+3} & 0 \\ 1 & 1 \end{bmatrix}.$$

Conformément à la théorie, celle-ci transforme la p-courbure en une matrice semblable triangulaire supérieure avec des zéros sur la diagonale.

```
> evalm(evalm(Inverse(P) mod p) &* pcurv &* P);
```

$$\begin{bmatrix} 0 & \dfrac{2\,x^3+2\,x^2+x+1}{x^9+4\,x^4} \\ 0 & 0 \end{bmatrix}$$

Et cette même matrice P réduit aussi notre système.

```
> P[A]:=evalm(evalm(Inverse(P) mod p) &* (A &* P - map(X->diff(X,x),P)));
```

$$P[A] := \begin{bmatrix} \dfrac{2\,x}{x^2+3} & \dfrac{3\,x^2+4}{4\,x+x^3} \\ 0 & \dfrac{3\,x+4}{x+x^2} \end{bmatrix}$$

1.7 Généralisations immédiates

Pour le problème de la factorisation de systèmes différentiels en caractéristique p, nous avons développé et implanté un algorithme basé sur des idées de [Pu95, Pu97] et [Ba01]. Nous avons aussi donné des comparaisons de stratégies et une analyse de complexité. Cet algorithme suit naturellement la classification des modules différentiels en caractéristique p de [Pu95] qui implique que la forme de Jordan de la p-courbure donne toutes les factorisations du système. Cependant, cette méthode peut entrainer des calculs dans des extensions algébriques. En profitant de plus d'éléments de l'eigenring comme dans [Ba01], nous proposons diverses heuristiques qui peuvent éviter de travailler dans ces extensions algébriques. Combiner ceci et les techniques de Giesbrecht et Zhang pourrait mener à d'autres améliorations.

Le fait, observé dans [Pu96], qu'une factorisation en caractéristique zéro se réduit modulo p pour presque tout p implique que notre algorithme peut être utilisé comme filtre : par exemple, si nous n'avons pas de facteurs à droite d'ordre 1 en caractéristique

p, alors nous n'avons pas de solutions exponentielles en caractéristique zéro. Un algorithme utilisant des méthodes modulaires pour calculer les solutions exponentielles en caractéristique zéro est développé dans le chapitre 2 (voir aussi [CH04]).

Nous allons maintenant montrer comment adapter l'algorithme de factorisation de systèmes différentiels développé dans les sections 1.3 à 1.6 à d'autres situations :

1.7.1 Factorisation « locale »

Dans cette sous-section, nous donnons les éléments nécessaires pour généraliser l'algorithme décrit dans les sections 1.3 à 1.6 au cas des systèmes à coefficients dans $\overline{\mathbb{F}}_p((x))$.

Soit $[A]$ un système différentiel avec $A \in \mathbb{M}_n(\overline{\mathbb{F}}_p((x)))$. Les notions de p-courbure et eigenring peuvent être définies de la même façon que dans les définitions 7 et 13. En remarquant que $\overline{\mathbb{F}}_p((x))$ est un corps C_1 ([Ja80, Definition 11.5, p. 649]), on déduit que la classification des modules différentiels en caractéristique p reste directement applicable dans ce cas (voir la sous-section 1.3.2).

Pour construire un algorithme comme nous l'avons fait dans les parties 1.3 à 1.6 dans le cas où $A \in \mathbb{M}_n(\overline{\mathbb{F}}_p(x))$, il ne nous reste donc plus qu'à savoir factoriser un polynôme à coefficients dans $\mathbb{F}_q((x))$ où $q = p^r$: ceci peut se faire grâce au théorème de Newton/Puiseux suivant (voir par exemple [Abh90, Lecture 12]) :

Théorème 3. *Soit $F(Y) = Y^n + a_{n-1}(x)\,Y^{n-1} + \cdots + a_0(x)$ un polynôme unitaire à coefficients dans $k((x))$ avec $k = \overline{\mathbb{F}}_p$ où $p > n$. Il existe $r \in \mathbb{N}^*$, non divisible par p, tel que $F(Y) = \prod_{i=1}^{n}(Y - \nu_i)$ avec $\nu_i \in k((x^{1/r}))$.*

Tous les ingrédients nécessaires sont maintenant réunis et, en appliquant ce théorème au polynôme caractéristique de la p-courbure, on obtient une généralisation immédiate de notre algorithme au cas des systèmes à coefficients dans $\mathbb{K}((x))$.

Remarque 9. *Dans le théorème 3, si F désigne le polynôme caractéristique de la p-courbure, alors les ν_i sont liés à ce qu'on appelle les* exposants généralisés *de l'opérateur différentiel. En fait, on peut définir une notion d'exposants généralisés en caractéristique p de la même manière qu'en caractéristique zéro et montrer qu'ils ne sont autres que les réductions mod p des exposants généralisés en caractéristique zéro : tout ceci sera détaillé dans le chapitre 2.*

1.7.2 Factorisation de systèmes d'équations aux différences

L'algorithme développé dans le cadre des systèmes différentiels linéaires $Y(x)' = A(x)\,Y(x)$ peut être généralisé au cas des systèmes aux différences linéaires $Y(x+1) = A(x)\,Y(x)$. Le corps différentiel $(k(x),')$ où $k = \mathbb{F}_q$ avec $q = p^r$ est remplacé par le corps aux différences $(k(x),\sigma)$ où σ est défini par $\sigma(x) = x + 1$ et $\sigma(f) = f$, pour tout $f \in k$. Le corps des constantes $\{a \in k \,|\, \sigma(a) = a\}$ est alors $k(x^p - x)$ (voir [PS97, Chapter 5]

ou [GZ03, Theorem 3.1]).

Tout comme dans le cas différentiel, il existe une notion naturelle de p-courbure :

Définition 15. *Soit $\sigma(Y) = AY$ avec $A \in \mathrm{GL}_n(k(x))$ ($k = \mathbb{F}_q$, $q = p^r$) un système d'équations aux différences linéaire. Sa* p-courbure *est le produit de matrices $A(x+p-1)\cdots A(x+1)\,A(x)$.*

Une classification des modules aux différences en caractéristique p (analogue à celle de [Pu95] ou [PS03, Ch. 13] dans le cas différentiel) est donnée dans [PS97, Ch. 5]. Cette classification implique là aussi que la forme de Jordan de la p-courbure donne toutes les factorisations du système aux différences. L'équation $\tau(y) := y^{(p-1)} + y^p = \lambda$ avec $\lambda \in k(x^p)$ considérée dans la sous-section 1.3.4 est ici remplacée par $\tau_1(u) := u(x+p-1)\cdots u(x+1)\,u(x) = \lambda$ avec $\lambda \in k(x^p - x)$. Dans le cas où la p-courbure est scalaire, le résultat de rationnalité du théorème 1 ne reste pas vrai ; en effet, si l'on suppose $p > n$ et que l'on essaye d'adapter la preuve, l'analogue de la solution $\mathrm{Tr}(A)/n$ de l'équation $\tau(y) = \lambda$ est ici la solution $\det(A)^{1/n}$ à l'équation $\tau_1(y) = \lambda$ et nous perdons donc la rationnalité de cette solution.

Nous pouvons aussi définir un eigenring (voir par exemple [Ba01, Bom01]) : soit $\sigma(Y) = AY$ un système d'équations aux différences linéaire avec $A \in \mathbb{M}_n(k(x))$. *L'eigenring $\mathcal{E}(A)$ de* $[A]$ est l'ensemble défini par

$$\mathcal{E}(A) \;=\; \{P \in \mathbb{M}_n(k(x)) \,|\, \sigma(P)\,A = A\,P\}.$$

Tous les éléments nécessaires au développement d'un algorithme analogue à celui que nous avons donné dans le cas différentiel sont donc réunis et l'algorithme suit naturellement. Notons encore que :

- les résultats de [Ba01] restent vrais dans le cas aux différences ([Ba01] est écrit dans le cadre général des équations pseudo-linéaires),
- l'algorithme de Giesbrecht et Zhang ([GZ03]) factorise tout polynôme de Ore donc en particulier les opérateurs aux différences.

1.8 Factorisation de systèmes d'équations aux dérivées partielles en caractéristique p

Dans cette partie, nous généralisons l'algorithme décrit précédemment pour factoriser certains systèmes d'équations aux dérivées partielles en caractéristique p : plus précisément, nous montrons comment ramener le problème à la réduction d'une combinaison linéaire à coefficients indéterminés de matrices qui commutent. Le fait d'être amené à réduire une combinaison linéaire à coefficients indéterminés de matrices pour factoriser un système d'équations aux dérivées partielles apparait déjà de manière naturelle lorsque l'on considère des systèmes à coefficients constants. En effet, soit $A_1, \ldots, A_m$ des matrices carrées à coefficients constants qui commutent entre elles deux

à deux. Considérons le système D-fini donné par les $\Delta_i := \partial_i - A_i$. Chaque système $\Delta_i(Y) = 0$ a pour solution $\exp(A_i\, x_i)\, C_i$ avec C_i vecteur constant ($\partial_i(C_i) = 0$) de sorte que $Y = \exp(A_1\, x_1 + \cdots + A_m\, x_m)$ est une solution commune à chaque $\Delta_i(Y) = 0$. Or si l'on veut calculer cette exponentielle de matrice, il nous faut réduire la combinaison linéaire à coefficients indéterminés $A_1\, x_1 + \cdots + A_m\, x_m$.

Concernant le problème de la factorisation de systèmes d'équations aux dérivées partielles en caractéristique zéro, récemment, Li, Schwarz et Tsarëv ont montré (voir [LST02, LST03]) comment adapter l'algorithme de Beke au cadre des systèmes d'équations aux dérivées partielles D-finis. Notons encore que dans sa thèse ([Wu04]), Wu décrira un algorithme de factorisation dans le cadre plus général des équations fonctionnelles.

1.8.1 Systèmes D-finis

Soit $\mathcal{F} = k(x_1, \ldots, x_m)$ le corps des fractions rationnelles en m variables $x_1, \ldots, x_m$ et à coefficients dans un corps k. Pour i appartenant à $\{1, \ldots, m\}$, notons $\partial_i := \frac{d}{dx_i}$ l'opérateur de dérivation par rapport à la i-ème variable. Soit $\Theta := \{\partial_1, \ldots, \partial_m\}$ le monoïde commutatif généré par les dérivations ∂_i.

En suivant la terminologie de [PS03, Appendix D], on dit alors que $(\mathcal{F}, \Theta)$ est un Θ-*corps* ou *corps différentiel partiel*.

Définition 16. *Soit $(\mathcal{F}, \Theta)$ un corps différentiel partiel.* L'anneau des opérateurs différentiels partiels à coefficients dans $\mathcal{F}$ *noté $\mathcal{F}[\Theta]$ est l'anneau (non-commutatif) des polynômes en les variables ∂_i satisfaisant les deux conditions suivantes :*

- *les ∂_i commutent entre eux deux à deux,*
- *pour tout $i \in \{1, \ldots, m\}$ et tout $a \in \mathcal{F}$, $\partial_i\, a = a\, \partial_i + \partial_i(a)$.*

Définition 17. *Un* système d'équations aux dérivées partielles *est donné par un ensemble fini d'éléments de l'anneau $\mathcal{F}[\Theta]$. À tout système S d'équations aux dérivées partielles, on associe l'idéal (à gauche) généré par les éléments de S que l'on note (S).*

Dans cette partie, tous les systèmes d'équations aux dérivées partielles que nous considérons sont supposés écrits sous la forme

$$\left\{ \begin{array}{lll} \Delta_1(y) = 0 & \text{avec} & \Delta_1 := \partial_1 - A_1, \\ \quad\vdots & & \\ \Delta_m(y) = 0 & \text{avec} & \Delta_m := \partial_m - A_m, \end{array} \right. \tag{1.7}$$

où les A_i sont des matrices de taille $n \in \mathbb{N}^*$ à coefficients dans $\mathcal{F}$ et les Δ_i commutent entre eux deux à deux : cette commutativité implique les relations suivantes, appelées *conditions d'intégrabilité*, sur les matrices A_i (voir par exemple [PS03, Appendix D]) :

$$\partial_i(A_j) - \partial_j(A_i) + A_i\, A_j - A_j\, A_i = 0, \tag{1.8}$$

pour tout i, j dans $\{1, \ldots, m\}$.

De tels systèmes sont des systèmes d'équations aux dérivées partielles *D-finis*.

Définition 18. *Un système S d'équations aux dérivées partielles est* D-fini *si l'idéal associé (S) est de dimension (différentielle) zéro.*

Si S est D-fini, alors le $\mathcal{F}$-espace vectoriel $\mathcal{F}[\Theta]/(S)$ est de dimension finie. Les systèmes D-finis correspondent donc à des $\mathcal{F}[\Theta]$-*modules* c'est-à-dire à des espaces vectoriels de dimension finie sur $\mathcal{F}$ qui sont des modules à gauche pour l'anneau $\mathcal{F}[\Theta]$ (Voir [PS03, Appendix D]). En terme de solutions, la D-finitude est équivalente au fait que les solutions du système ne dépendent que d'un nombre fini de constantes.

Remarque 10. *Lorsque k est un corps de caractéristique p, alors nous supposerons toujours que $p > n$ ce qui assure l'existence de vecteurs cycliques (voir [CK02, Section 6]).*

Étant donné un système d'équations aux dérivées partielles quelconque S, il existe des algorithmes permettant de tester si le système est D-fini et, si oui, de l'écrire sous la forme (1.7). Par exemple, ceci peut se faire en calculant une *base de Janet* (base aussi appelée *base involutive* dans la littérature) du système (voir [Ja20, Ja29, HS02, BCGPR03]). Ces bases de Janet peuvent être vues comme une sorte d'analogue des bases de Groebner (non réduites). L'intérêt est qu'une base de Janet fournit une base du quotient $\mathcal{F}[\Theta]/(S)$: la finitude de cette base est alors équivalente au fait que le système soit D-fini et les A_i s'obtiennent en calculant l'action des ∂_i sur la base du quotient. En particulier, la dimension n des A_i est égale au cardinal de cette base.

1.8.2 Factorisation et eigenrings

De la même manière que nous l'avons fait dans la sous-section 1.1.5, nous pouvons définir ici les notions de *systèmes décomposables et réductibles* et en déduire ce que nous entendons par *factoriser un système d'équations aux dérivées partielles.*

Définition 19. *Soit $(\mathcal{F}, \Theta)$ un corps différentiel partiel et considérons un système d'équations aux dérivées partielles S.*
On dit que le système S est décomposable *(sur $\mathcal{F}$) s'il existe une matrice P inversible à coefficients dans $\mathcal{F}$ telle que pour tout i dans $\{1, \ldots, m\}$, $P^{-1}(A_i P - \partial_i(P))$ soit diagonale par blocs. Un système qui n'est pas décomposable est dit* indécomposable.
On dit que le système S est réductible *(sur $\mathcal{F}$) s'il existe une matrice P inversible à coefficients dans $\mathcal{F}$ telle que pour tout i dans $\{1, \ldots, m\}$, $P^{-1}(A_i P - \partial_i(P))$ soit triangulaire par blocs. Un système qui n'est pas réductible est dit* irréductible.

Définition 20. *Soit $(\mathcal{F}, \Theta)$ un corps différentiel partiel et considérons un système d'équations aux dérivées partielles S.* Factoriser le système *S signifie, soit trouver une matrice P inversible à coefficients dans $\mathcal{F}$ telle que pour tout i dans $\{1, \ldots, m\}$, $P^{-1}(A_i P - \partial_i(P))$ soit diagonale ou triangulaire par blocs, soit montrer qu'une telle matrice P n'existe pas. Autrement dit, cela signifie décider si S est décomposable ou indécomposable, réductible ou irréductible, et dans le cas décomposable ou réductible, trouver une matrice P réduisant ou décomposant simultanément tous les systèmes $\partial_i(Y) = A_i Y$.*

Factoriser un système d'équations aux dérivées partielles signifie donc factoriser *simultanément* les systèmes $\partial_i(Y) = A_i\,Y$. En particulier, nous voyons déjà que si l'un de ces systèmes est irréductible sur $\mathcal{F}$, alors le système S est lui aussi irréductible sur $\mathcal{F}$.

Dans le cas différentiel classique, nous avons caractérisé l'eigenring d'un système $\Delta_A(Y) = 0$ comme l'ensemble des endomorphismes différentiels qui commutent avec Δ_A : nous prenons ici cette caractérisation pour définition.

Définition 21. *Soit $(\mathcal{F}, \Theta)$ un corps différentiel partiel et considérons un système d'équations aux dérivées partielles S. L'*eigenring du système *est*

$$\mathcal{E}(S) := \{P \in \mathbb{M}_n(\mathcal{F}) \,|\, \forall i \in \{1, \ldots, m\},\, P\,\Delta_i = \Delta_i\,P\}.$$

Si l'on note $\mathcal{E}_i(S) := \{P \in \mathbb{M}_n(\mathcal{F}) \,|\, P\,\Delta_i = \Delta_i\,P\}$ alors on a

$$\mathcal{E}(S) = \cap_{i=1}^{n} \mathcal{E}_i(S).$$

Ces ensembles $\mathcal{E}_i(S)$ sont appelés les eigenrings partiels.

De manière analogue à ce qui se passe dans le cas différentiel ordinaire, la connaissance d'un élément non-trivial dans l'eigenring donne immédiatement une factorisation du système.

1.8.3 Systèmes d'é.d.p. en caractéristique p

Considérons maintenant le corps différentiel partiel $(\mathbb{K}, \Theta)$ de caractéristique $p > n$ où $\mathbb{K} := k(x_1, \ldots, x_m)$ avec $k = \mathbb{F}_q$, $q = p^r$. Comme nous avons ici plusieurs dérivations, nous définissons des p-courbures partielles.

Définition 22. *Soit S un système d'équations aux dérivées partielles à coefficients dans $(\mathbb{K}, \Theta)$. Les* p-courbures partielles du système S *sont les opérateurs Δ_i^p, pour $i \in \{1, \ldots, m\}$, agissant sur $\mathbb{K}^n$.*

Notons que dans [Ka82], Katz définit aussi une notion de p-courbure dans le cas de plusieurs dérivations et remarque les liens entre cette p-courbure et l'eigenring du système.

Lemme 13. *Soit S un système d'équations aux dérivées partielles à coefficients dans $(\mathbb{K}, \Theta)$. Les p-courbures partielles Δ_i^p commutent entre elles (deux à deux) et appartiennent à l'eigenring $\mathcal{E}(S)$. De plus le polynôme minimal (et caractéristique) de chaque Δ_i^p est à coefficients dans $k(x_1^p, \ldots, x_m^p)$.*

Démonstration. Le fait que les p-coubures partielles commutent et appartiennent à l'eigenring découle directement du fait que les Δ_i commutent entre eux deux à deux. Soit $\chi_1(T) := T^d + \sum_{i=0}^{d-1} f_i\,T^i$, avec $d \in \mathbb{N} et d \leq n$, le polynôme minimal de $D_1 := \Delta_1^p$. On sait que χ_1 est à coefficients dans le corps des constantes de $(k(x_1, \ldots, x_m), \partial_1)$, *i. e.*, $k(x_2, \ldots, x_m)(x_1^p)$. Pour tout $j \in \{2, \ldots, m\}$, on a $\Delta_j(\chi_1(D_1)) = 0$. En utilisant le fait que Δ_j vérifie la condition de Leibniz, on obtient alors $\sum_{i=0}^{d-1} \partial_j(f_i)\,D_1^i = 0$. Maintenant, comme χ_1 est le polynôme minimal de D_1, il vient $\partial_j(f_i) = 0$ pour tout $i \in \{0, \ldots, d-1\}$ et pour tout $j \in \{2, \ldots, m\}$; d'où le résultat. □

Le corps de base $\mathbb{K} = k(x_1, \ldots, x_m)$ avec $k = \mathbb{F}_q$ et $q = p^r$ vérifie $\mathbb{K}^{(p)} := \{f^p \mid f \in \mathbb{K}\} = k(x_1^p, \ldots, x_m^p)$. Nous avons donc $[\mathbb{K} : \mathbb{K}^{(p)}] = m\,p \neq p$ de sorte que la première hypothèse sur le corps de base, nécessaire pour que la classification de van der Put s'applique directement, n'est pas satisfaite. Cependant, certaines preuves que nous avons ré-écrites d'une manière élémentaire et constructive peuvent s'adapter ici et nous permettent de généraliser les résultats de la section 1.3.

Intéressons nous tout d'abord au lemme de Cartier (voir Lemme 3 dans le cas différentiel ordinaire) dans le cadre des équations aux dérivées partielles (voir [Ka70, Theorem 5.1]).

Proposition 11 (Cartier). *Soit S un système d'équations aux dérivées partielles à coefficients dans $(\mathbb{K}, \Theta)$. Les p-coubures partielles Δ_i^p sont toutes nulles si, et seulement si, le système S possède une base de solutions rationnelles.*

Ici, S possède une base de solutions rationnelles signifie qu'il existe une base de solutions rationnelles commune à tous les $\Delta_i(Y) = 0$. Autrement dit, il existe $P \in \mathrm{GL}_n(\mathbb{K})$ telle que, pour tout $i \in \{1, \ldots, m\}$, $P^{-1}\,(A_i\,P - \partial_i(P)) = 0$.

Démonstration. L'implication « $\Leftarrow$ » étant immédiate, il nous suffit de montrer « $\Rightarrow$ ». Considérons tout d'abord le corps différentiel $(k(x_2, \ldots, x_m)(x_1), \partial_1)$ ayant pour corps de constantes $k(x_2, \ldots, x_m)(x_1^p)$, et regardons Δ_1 comme un opérateur différentiel agissant sur $k(x_2, \ldots, x_m)(x_1)^n$; puisqu'il vérifie $\Delta_1^p = 0$, le lemme de Cartier nous permet d'affirmer l'existence de $P_1 \in \mathrm{GL}_n(k(x_2, \ldots, x_m)(x_1))$ telle que $P_1^{-1}\,\Delta_1\,P_1 = \partial_1$. Pour $i \in \{1, \ldots, m\}$, posons $\tilde{\Delta}_i = P_1^{-1}\,\Delta_i\,P_1 := \partial_i - B_i$ pour des matrices B_i à coefficients dans $k(x_1, \ldots, x_m)$. Les conditions d'intégrabilité impliquent que pour tout $i \in \{1, \ldots, m\}$, $\partial_1(B_i) = 0$ de sorte que les B_i sont à coefficients dans $k(x_2, \ldots, x_m)(x_1^p)$. Utilisons maintenant l'hypothèse $\tilde{\Delta}_2^{\,p} = 0$ et appliquons le lemme de Cartier à $\tilde{\Delta}_2$: il existe $P_2 \in \mathrm{GL}_n(k(x_2, \ldots, x_m)(x_1^p))$ telle que $P_2^{-1}\,\tilde{\Delta}_2\,P_2 = \partial_2$. De plus $P_2 \in \mathrm{GL}_n(k(x_2, \ldots, x_m)(x_1^p))$ implique que ∂_1 commute avec P_2 et donc $P_2^{-1}\,\tilde{\Delta}_1\,P_2 = P_2^{-1}\,\partial_1\,P_2 = \partial_1$. En continuant ce raisonnement on trouve finalement une matrice $P = P_1 \cdots P_m$ inversible à coefficients dans $k(x_1, \ldots, x_m)$ telle que pour tout $i \in \{1, \ldots, m\}$, $P^{-1}\,\Delta_i\,P = \partial_i$ d'où le résultat. □

Cette preuve exhibe un algorithme pour le calcul d'une matrice fondamentale de solutions rationnelles d'un système d'équations aux dérivées partielles :

- Pour $i = 1$ à m, poser $A_i^{[1]} := A_i$.
- Pour $i = 1$ à m :
  - Calculer une matrice fondamentale P_i de solutions rationnelles du système différentiel (à une variable) $\partial_i(Y) = A_i^{[i]}\,Y$,
  - pour $j = 1$ à m, calculer $A_j^{[i+1]} := P_i^{-1}\,(A_j^{[i]}\,P_i - \partial_j(P_i))$.
- Retourner $P_1 \cdots P_m$.

Remarque 11. *Dans le cas où l'une des p-courbures partielles est nulle, alors, après*

un changement de base, le système (1.7) s'écrit

$$\left\{\begin{array}{lll} \Delta_1(y)=0 & \textit{avec} & \Delta_1 := \partial_1, \\ \quad\vdots & & \\ \Delta_m(y)=0 & \textit{avec} & \Delta_m := \partial_m - A_m, \end{array}\right. \tag{1.9}$$

de sorte que les conditions d'intégrabilité (1.8) impliquent $\partial_1(A_j) = 0$ *pour tout* $j \in \{2,\dots,m\}$. *On peut donc en déduire que le système ne dépend plus de* x_1 *mais seulement de* x_1^p.

Regardons maintenant le cas où toutes les p-courbures partielles Δ_i^p sont scalaires c'est-à-dire, pour tout $i \in \{1,\dots,m\}$, $\Delta_i^p = \lambda_i\, I_n$ avec $\lambda_i \in k(x_1^p,\dots,x_m^p)$ (voir Lemme 13).

Considérons tout d'abord individuellement le système $\partial_1(Y) = A_1\, Y$ à coefficients dans le corps différentiel $\mathbb{K} = k(x_2,\dots,x_m)(x_1)$ muni de la dérivation ∂_1 et ayant pour corps de constantes $k(x_2,\dots,x_m)(x_1^p)$. Notre résultat de rationnalité (Théorème 1) est ici indispensable. En effet, si l'on considère une équation $\partial_1^{p-1}(y) + y^p = c$ avec $c \in k(x_2,\dots,x_m)(x_1^p)$, alors en utilisant la décomposition en éléments simples nous ne pouvons que démontrer l'existence d'une solution dans $\overline{k(x_2,\dots,x_m)}(x_1)$ (voir le lemme 6 et sa preuve). Cependant le résultat du théorème 1 peut aussi être utilisé ici et nous donne une solution dans $k(x_2,\dots,x_m)(x_1)$. Plus précisément, si $\Delta_1^p = -\lambda_1\, I_n$, alors $\mu_1 = -\mathrm{Tr}(A_1)/n$ vérifie $\partial_1^{p-1}(\mu_1) + \mu_1^p = \lambda_1$ et le système $\partial_1(Y) = A_1\, Y$ est donc équivalent sur $\mathbb{K}$ à $[\mu_1\, I_n]$.

Proposition 12. *Soit* S *un système d'équations aux dérivées partielles à coefficients dans* $(\mathbb{K},\Theta)$. *Les* p*-coubures partielles* Δ_i^p *sont toutes scalaires si, et seulement si, le système* S *est équivalent sur* $\mathbb{K}$ *à un « système diagonal ». Autrement dit, pour tout* $i \in \{1,\dots,m\}$, $\Delta_i^p = \lambda_i\, I_n$ *avec* $\lambda_i \in k(x_1^p,\dots,x_m^p)$ *si, et seulement si, il existe* $P \in \mathrm{GL}_n(\mathbb{K})$ *telle que pour tout* $i \in \{1,\dots,m\}$, $P^{-1}\,(A_i\, P - \partial_i(P)) = \mu_i\, I_n$ *avec* $\mu_i \in \mathbb{K}$.

Démonstration. Supposons $m = 2$ ce qui n'enlève rien à la généralité de la preuve. Considérons donc un système d'équations aux dérivées partielles $(\partial_1 - A_1, \partial_2 - A_2)$ tel que $\Delta_1^p = \lambda_1\, I_n$ et $\Delta_2^p = \lambda_2\, I_n$ avec $\lambda_1,\ \lambda_2 \in k(x_1^p, x_2^p)$. Posons $\mu_1 = \mathrm{Tr(A_1)}/n$ et $\mu_2 = \mathrm{Tr(A_2)}/n$ et considérons le système d'équations aux dérivées partielles $(\Delta_1 - \mu_1\, I_n, \Delta_2 - \mu_2\, I_n)$. Par construction ce système possède des p-courbures partielles nulles. De plus la condition d'intégrabilité pour ce nouveau système est satisfaite : en effet, après simplifications, cette condition s'écrit $\partial_1(\mu_2) = \partial_2(\mu_1)$ ce qui est équivalent à $\mathrm{Tr}(\partial_1(A_2)) = \mathrm{Tr}(\partial_2(A_1))$ ou encore, d'après (1.8), à $\mathrm{Tr}(A_2\, A_1) = \mathrm{Tr}(A_1\, A_2)$. La proposition 11 nous affirme alors l'existence d'une matrice P inversible et à coefficients dans $\mathbb{K}$ telle que $P^{-1}\,(\,(A_1 + \mu_1\, I_n)\, P - \partial_i(P)) = P^{-1}\,(\,(A_2 + \mu_2\, I_n)\, P - \partial_i(P)) = 0$; d'où le résultat. □

Ici aussi nous pouvons donner une « formule de Katz », analogue à celle donnée dans la sous-section 1.3.5, pour le calcul d'une matrice fondamentale de solutions rationnelles

lorsque les p-courbures partielles sont toutes nulles. Pour ceci, pour $i \in \{1, \ldots, m\}$, définissons

$$\mathrm{Pr}_i \ : \ \mathbb{K}^n \to \mathbb{K}^n, \ v \mapsto \sum_{k=0}^{p-1} \frac{(-x_i)^k}{k!} \, \Delta_i^k(v).$$

On vérifie alors que :

- pour tout $v \in \mathbb{K}^n$, $\Delta_i(\mathrm{Pr}_i(v)) = -\,(-x_i)^{p-1} \, \Delta_i^p(v)$ de sorte que Pr_i envoye $\ker(\Delta_i^p)$ sur $\ker(\Delta_i)$,
- pour tout i, $j \in \{1, \ldots, m\}$ tels que $i \neq j$, $\Delta_j(\mathrm{Pr}_i(v)) = \mathrm{Pr}_i(\Delta_j(v))$ de sorte que les Pr_i commutent entre eux deux à deux.

Posons maintenant $\mathrm{Pr} := \prod_{i=1}^m \mathrm{Pr}_i$. Cet opérateur de $\mathbb{K}^n$ dans $\mathbb{K}^n$ satisfait alors la propriété suivante : pour tout $i \in \{1, \ldots, m\}$, si $\Delta_i^p(v) = 0$, alors $\Delta_i(\mathrm{Pr}(v)) = 0$.

D'après [Ka70, §5, p. 191], cette formule pour $\mathrm{Pr}(v)$ peut être développée et on obtient ainsi :

$$\mathrm{Pr}(v) = \sum_{\omega} \prod_{i=1}^m \frac{(-x_i)^{\omega_i}}{\omega_i!} \prod_{i=1}^m \Delta_i^{\omega_i}(v),$$

où la somme est prise sur tous les r-uplets $\omega = (\omega_1, \ldots, \omega_r)$ d'entiers tels que $0 \leq \omega_i \leq p-1$.

Ceci nous donne donc une formule explicite pour le calcul d'une matrice fondamentale de solutions rationnelles d'un système d'équations aux dérivées partielles dans le cas où les p-courbures partielles Δ_i^p sont toutes nulles et où $(0, \ldots, 0)$ n'annule pas le dénominateur du système (voir la discussion suivant la remarque 5).

Remarque 12. *Dans [Ka70], Katz donne aussi un analogue de la « formule de Taylor »* $\mathrm{T_a}$ *(utilisée dans la preuve du lemme 9 de la sous-section 1.3.5) dans le cas de plusieurs dérivations.*

1.8.4 Factorisation de systèmes d'é.d.p. en caractéristique p

Soit S un système d'équations aux dérivées partielles à coefficients dans $(\mathbb{K}, \Theta)$ avec $\mathbb{K} = k(x_1, \ldots, x_m)$ et $k = \mathbb{F}_q$ où $q = p^r$. On sait que pour factoriser individuellement le système $\partial_i(Y) = A_i \, Y$, alors nous pouvons appliquer l'algorithme développé dans les sections 1.3 à 1.6. Pour ce faire, nous utilisons la p-courbure partielle Δ_i^p ainsi que l'eigenring partiel $\mathcal{E}_i(S)$. Tout ceci peut se faire puisqu'au cours de l'algorithme, nous ne faisons rien d'autre que de l'algèbre linéaire, soit sur la p-courbure, soit sur un élément de l'eigenring. Maintenant, si nous voulons factoriser le système d'équations aux dérivées partielles S, nous devons factoriser simultanément les $\partial_i(Y) = A_i \, Y$. Nous sommes donc amenés à réduire simultanément les p-courbures partielles Δ_i^p qui commutent d'après le lemme 13. Le problème de la réduction simultanée de matrices qui commutent sera étudié dans la sous-section-suivante.

Comme nous l'avons fait dans le cas différentiel ordinaire, nous allons tout d'abord chercher à décomposer le système puis nous montrerons comment réduire les blocs

indécomposables. La première étape consiste donc à calculer une *décomposition isotypique simultanée* du système.

Définition 23. *Soit S un système d'équations aux dérivées partielles. Une* décomposition isotypique simultanée *de S est donnée par une matrice inversible P telle que :*

- *pour tout i, $P^{-1}\,(A\,P - \partial_i(P)) = \bigoplus_{j=1}^{l_i}[B_i^{[j]}]$, et*
- *pour tout i, la p-courbure partielle de chaque système $\partial_i(Y) = B_i^{[j]}\,Y$ a un polynôme caractéristique de la forme $F_{i,j}^{m_{i,j}}$ avec $F_{i,j}$ irréductible, $F_{i,j} \neq F_{i,k}$ si $j \neq k$ et $m_{i,j} \geq 1$.*

Exemple 12. *Prenons $\mathbb{K} := \mathbb{F}_p(x_1, x_2)$ avec $p = 3$ et considérons le système d'équations aux dérivées partielles D-fini donné par ($\partial_1 - A_1$,$\partial_2 - A_2$) où A_1 et A_2 sont les deux matrices suivantes :*

$$A_1 = \begin{pmatrix} 1 & x_1\,x_2 \\ 0 & 1 \end{pmatrix},$$

$$A_2 = \begin{pmatrix} \frac{x_1 - 2x_1^3x_2f_2(x_2) - f_3(x_2)x_1 + x_1x_2(f_1(x_2) + f_2(x_2)x_1^2)}{x_1x_2} & \frac{1}{2}f_2(x_2)x_2x_1^4 + \frac{1}{2}f_3(x_2)x_1^2 + f_4(x_2) \\ -\frac{2f_2(x_2)}{x_2} & f_1(x_2) + f_2(x_2)x_1^2 \end{pmatrix},$$

où f_1, f_2, f_3 et f_4 sont des fonctions de la variable x_2.

Cas 1 : *regardons tout d'abord le cas où :*

$$\{f_1(x_2) = x_2^4,\ f_2(x_2) = x_2,\ f_3(x_2) = x_2^6,\ f_4(x_2) = 2\,x_2^6 + 2\,x_2^4\}.$$

En calculant les p-courbures partielles et en factorisant leur polynôme caractéristique, nous trouvons :

- $\chi(\Delta_1^p)(T) = (T+1)^2$,
- $\chi(\Delta_2^p)(T) = T^2 + (2\,x_2^{12} + x_2^3 + 2\,x_2^{15})\,T + x_2^{15} + x_2^{18} + 2\,x_2^{27} + x_2^{24}$.

Le fait que $\chi(\Delta_2^p)$ soit irréductible implique que les p-courbures partielles ne peuvent pas être réduites simultanément et par conséquent, le système de départ est irréductible sur $\overline{\mathbb{F}}_3(x_1, x_2)$.

Cas 2 : *maintenant, si*

$$\{f_1(x_2) = 2\,x_2,\ f_2(x_2) = 0,\ f_3(x_2) = 2x_2^6 + x_2,\ f_4(x_2) = x_2 + x_2^2\}.$$

alors, on trouve :

- $\chi(\Delta_1^p)(T) = (T+1)^2$,
- $\chi(\Delta_2^p) = (T + 2 + x_2^3 + x_2^{15})(T + 2\,x_2^3)$,

de sorte que le système est diagonalisable. L'algorithme IsoDecSim *qui sera donné dans la sous-section suivante nous renvoie la matrice suivante.*

```
> P:=IsoDec_Simult(S,n,p);
```

$$P := \begin{bmatrix} 1 & \dfrac{2\,({x_2}^2 + {x_2}^5 + {x_2}^{11} + 2\,{x_2}^4\,{x_1}^2 + {x_2}^{12} + {x_2}^7 + 2\,{x_1}^2\,x_2 + {x_1}^2\,{x_2}^{16})}{2\,{x_2}^3 + 2 + {x_2}^{15}} \\ 0 & 1 \end{bmatrix}.$$

Nous pouvons maintenant vérifier que cette matrice décompose simultanément les deux systèmes :

```
> evalm(map(X->Normal(X) mod p,evalm((Inverse(P) mod p) &* (A_1 &* P
- map(X->diff(X,x_1),P)))));
```

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

```
> evalm(map(X->Normal(X) mod p,evalm((Inverse(P) mod p) &* (A_2 &* P
- map(X->diff(X,x_2),P)))));
```

$$\begin{bmatrix} \dfrac{{x_2}^6 + 2\,{x_2}^2 + 2\,x_2 + 1}{x_2} & 0 \\ 0 & 2\,x_2 \end{bmatrix}.$$

En fait d'une manière générale, nous pouvons voir que la factorisation du système de départ est la suivante :

- *Si $f_2(x_2) \neq 0$, alors le système est irréductible,*
- *Si $f_2(x_2) = 0$, alors le système est décomposable.*

Nous sommes maintenant confrontés au cas où les m p-courbures partielles Δ_i^p ont toutes un polynôme caractéristique de la forme $F_i^{m_i}$ avec F_i irréductible et $m_i > 1$. Ici encore nous pouvons essayer de décomposer simultanément les Δ_i^p : cependant, le problème va être le même que dans le cas différentiel ordinaire à savoir que la matrice P qui va décomposer simultanément les Δ_i^p ne va en général pas décomposer les systèmes différentiels $\partial_i(Y) = A_i\,Y$. Par conséquent les mêmes alternatives que dans le cas différentiel s'offrent à nous :

- l'heuristique consistant à choisir un élément au hasard dans l'eigenring $\mathcal{E}(S)$ et à décomposer le système en utilisant cet élément (la proposition 8 reste vraie ici),
- calculer des idempotents dans l'eigenring, décomposer le système et recommencer (comme dans [GZ03]).

Une fois que l'on a calculé une décomposition maximale, il ne nous reste plus qu'à traiter le cas indécomposable. Pour ceci il nous suffit d'utiliser la méthode décrite dans la sous-section suivante pour triangulariser simultanément les Δ_i^p sur $\mathbb{K}$; ceci revient à appliquer la méthode donnée dans la preuve de la proposition 10 de la section 1.6 à la combinaison linéaire à coefficients indéterminés $D := \sum_{i=1}^{m} t_i \, \Delta_i^p$: si F^m est le polynôme caractéristique (et minimal) de D, alors on forme la base adaptée au drapeau de $\mathbb{K}^n$ formé par les $E_i := \ker(F^i(D))$ pour $i = 1, \ldots, m-1$ et la matrice de chaque système $\partial_i(Y) = A_i \, Y$ dans cette nouvelle base est triangulaire par blocs.

1.8.5 Réduction simultanée de matrices qui commutent

Dans cette sous-section, nous nous intéressons au problème de la réduction simultanée de matrices qui commutent. Ce qui est exposé ici peut être considéré comme un travail indépendant du reste de ce chapitre et reste un sujet de nos recherches actuelles.

Soit $M_1, \ldots, M_s$ des matrices carrées de taille n à coefficients dans un corps K quelconque. Supposons que $M_1, \ldots, M_s$ commutent, *i. e.*, $[M_i, M_j] := M_i \, M_j - M_j \, M_i = 0$ pour tout i, j dans $\{1, \ldots, s\}$. Le but dans cette sous-section est de montrer comment réduire simultanément, quand cela est possible, les M_i c'est-à-dire trouver une matrice P inversible telle que, pour tout $i \in \{1, \ldots, s\}$, les $P^{-1} \, M_i \, P$ possèdent une forme simple donnée : nous nous intéressons ici essentiellement aux formes diagonales et triangulaires.

Rappelons tout d'abord quelques résultats connus :

- les M_i sont simultanément triangularisables sur $\overline{K}$, *i. e.*, il existe $P \in \mathrm{GL}_n(\overline{K})$, telle que pour tout $i \in \{1, \ldots, s\}$, $P^{-1} \, M_i \, P$ soit triangulaire supérieure (voir par exemple [CO68, Corollaire 1, p. 292]).
- si les M_i sont toutes diagonalisables sur K, alors elles sont simultanément diagonalisables sur K, *i. e.*, il existe $P \in \mathrm{GL}_n(\overline{K})$ telle que, pour tout $i \in \{1, \ldots, s\}$, $P^{-1} \, M_i \, P$ soit diagonale (voir par exemple [CO68, Théorème 19.6, p. 291]).

Lorsque les matrices M_i possèdent des polynômes caractéristiques scindés sur K, alors on a le résultat suivant :

Proposition 13 ([CO68], Proposition 19.8, p. 293). *Soit $\mathcal{M} = \{M_1, \ldots, M_s\}$ un ensemble de matrices à coefficients dans K qui commutent entre elles deux à deux et dont le polynôme caractéristique est scindé sur K. Il existe une matrice $P \in \mathrm{GL}_n(K)$ telle que, pour tout élément U de $\mathcal{M}$,*

$$P^{-1} \, U \, P = \begin{pmatrix} U_1 & & & \\ & U_2 & & \\ & & \ddots & \\ & & & U_r \end{pmatrix},$$

avec pour tout $i \in \{1, \ldots, r\}$,

$$U_i = \lambda_i(U) \, I_{n_i} + N_i(U),$$

où les U_i sont de dimension n_i ($\sum_{i=1}^{r} n_i = n$), où $\lambda_i(U)$ est un scalaire appartenant à K, et où $N_i(U)$ est une matrice trigonale supérieure nilpotente.

Nous allons maintenant présenter une sorte de généralisation de ce résultat dans le cas où les polynômes caractéristiques des matrices ne sont pas scindés sur K. Pour ce faire, nous définissons une notion de *décomposition isoptypique simultanée.*

Définition 24. *Soit $\mathcal{M} = \{M_1, \ldots, M_s\}$ une famille de matrices à coefficients dans K qui commutent entre elles deux à deux. Une* décomposition isotypique simultanée de $\mathcal{M}$ *est la donnée d'une matrice P inversible et à coefficients dans K telle que :*

- *pour tout $i \in \{1, \ldots, m\}$, $P^{-1} M_i P = \bigoplus_{j=1}^{d_i} M_{i,j}$ avec $\chi(M_{i,j}) = F_{i,j}^{m_{i,j}}$ où les $F_{i,j}$ sont irréductibles, et*
- *il existe $i \in \{1, \ldots, m\}$, tel que $P^{-1} M_i P$ soit la décomposition isotypique de M_i,* i. e., $\operatorname{pgcd}(F_{i,j}, F_{i,k}) = 1$ *pour $j \neq k$.*

Nous allons voir dans ce qui suit comment calculer, de manière effective, une décomposition isotypique simultanée de $\{M_1, \ldots, M_s\}$.

La première idée qui vient à l'esprit pour réduire simultanément des matrices qui commutent est de réduire l'une des matrices et de regarder l'effet de cette réduction sur les autres.

Lemme 14. *Soit $\{M_1, \ldots, M_s\}$ une famille de matrices à coefficients dans K qui commutent entre elles deux à deux. Supposons que pour un certain $i \in \{1, \ldots, s\}$, $\chi(M_i) = F_1 \cdots F_{b_i}$ avec les F_k irréductibles et premiers entre eux. Alors on peut effectivement trouver P inversible et à coefficients dans K telle que pour tout $j \in \{1, \ldots, s\}$, $P^{-1} M_j P$ soit diagonale par blocs.*

Démonstration. Nous savons que si $\chi(M_i) = F_1 \cdots F_{b_i}$ avec les F_k irréductibles et premiers entre eux, alors $\mathbb{K}^n = \bigoplus_{j=1}^{b_i} \ker(F_j(M_i))$. Maintenant, comme tous les M_k commutent avec M_i, $\ker(F_j(M_i))$ est stable sous M_k d'où le résultat. □

Cette méthode donnant une décomposition par blocs de chacune des matrices ne donne pas forcément une décomposition isotypique simultanée au sens où nous l'avons définie. Cependant, en appliquant récursivement ce procédé aux blocs obtenus, nous en déduisons un algorithme, IsoDecSim, calculant une décomposition isotypique simultanée de $\{M_1, \ldots, M_s\}$ sur K.

Procédure IsoDecSim
Entrée : $\mathcal{M} = \{M_1, \ldots, M_s\}$ (avec $M_i \in \mathbb{M}_n(K)$) s matrices qui commutent et la dimension n des M_i.
Sortie : $P \in \mathbb{M}_n(K)$ donnant une décomposition isotypique simultanée des M_i sur K.
1 - Si $\mathcal{M} = \emptyset$, alors retourner $P = I_n$.
2 - Si M_1 est scalaire ($M_1 = \lambda\, I_n$ pour λ dans K), retourner IsoDecSim($\{M_2, \ldots, M_s\}$,n).
3 - Soit $\Pi_1 := \chi(M_1)$ le polynôme caractéristique de M_1.
4 - Si $\Pi_1 = F_1^{m_1} \cdots F_d^{m_d}$ avec les F_i irréductibles sur K et premiers entre eux, alors :
  4a - Si $d = 1$ et $m_1 = 1$, retourner I_n.
  4b - Sinon :
    - Si $d = 1$, alors retourner IsoDecSim($\{M_2, \ldots, M_s\}$,n), sinon :
    - Soit P_0 telle que $P_0^{-1}\, M_1\, P_0 = \bigoplus_{j=1}^{d} M_{1,j}$.
    - Pour $i = 2, \ldots, s$, poser $B_i = P_0^{-1}\, M_i\, P_0$ (B_i est d-blocs diagonale).
    - Pour $i = 2, \ldots, s$, pour $j = 1, \ldots, d$, soit $B_{i,j}$ le j-ème bloc diagonal de taille $m_j \cdot \deg(F_j)$ dans B_i.
    - Pour $j = 1, \ldots, d$, poser $P_1[j] :=$ IsoDecSim($\{B_{2,j}, \ldots, B_{d,j}\}$, $m_j \cdot \deg(F_j)$).
    - $P_1 := \bigoplus_{j=1}^{d} P_1[j]$.
    - Retourner $P_0\, P_1$.

Nous allons maintenant proposer une autre méthode pour calculer cette décomposition isotypique simultanée.

Considérons la matrice

$$M := t_1\, M_1 + \cdots + t_s\, M_s, \tag{1.10}$$

à coefficients dans $K[t_1, \ldots, t_s]$.

Pour tout $i \in \{1, \ldots, s\}$, il existe un unique couple de matrices (S_i, N_i) avec S_i semi-simple (c'est-à-dire diagonalisable sur $\overline{K}$) et N_i nilpotente tel que :
– $M_i = S_i + N_i$,
– $[S_i, N_i] = 0$.
Une telle décomposition $M_i = S_i + N_i$ s'appelle *décomposition* SN *de* M_i.

Remarque 13. *Les valeurs propres de M_i dans $\overline{K}$ coïncident avec les valeurs propres de S_i dans $\overline{K}$. Autrement dit, M_i et S_i ont le même polynôme caractéristique.*

Lemme 15. *Si l'on note*

$$\begin{aligned} S &= t_1\, S_1 + \cdots + t_s\, S_s, \\ N &= t_1\, N_1 + \cdots + t_s\, N_s, \end{aligned}$$

alors S est semi-simple, N est nilpotente et $[S, N] = 0$. Par conséquent $M = S + N$ est la décomposition SN *de la matrice $M = \sum_{i=1}^{s} t_i\, M_i$.*

Démonstration. On sait (voir [CO68, Théorème 19.6, p. 294] ou [Le94]) que pour tout i, S_i et N_i sont des polynômes en M_i. Par conséquent, comme les M_i commutent entre elles deux à deux, on a

$$[S_i, S_j] = [N_i, N_j] = [N_i, S_j] = 0,$$

pour tout i, j dans $\{1, \ldots, s\}$. Les s matrices $S_1, \ldots, S_s$ commutent donc entre elles deux à deux et sont semi-simples. On sait alors qu'elles peuvent être diagonalisées simultanément sur $\overline{K}$ c'est-à-dire qu'il existe P inversible et à coefficients dans $\overline{K}$ telle que, pour tout $i \in \{1, \ldots, s\}$, $P^{-1} S_i P$ est diagonale. Le fait que S soit semi-simple suit immédiatement. Notons respectivement $u_1, \ldots, u_s$ les indices de nilpotence de $N_1, \ldots, N_s$ c'est-à-dire, pour tout $i \in \{1, \ldots, s\}$, $N_i^{u_i} = 0$ et $N_i^k \neq 0$ pour tout $k < u_i$. Alors, un calcul direct montre que N est nilpotente d'indice au plus $u_1 + \cdots + u_s$. Enfin, l'égalité $[S, N] = 0$ est claire puisque, pour tout $i \in \{1, \ldots, s\}$, $[S_i, N_i] = 0$. □

Corollaire 2. *Avec les notations précédentes, soit $(v_1, \ldots, v_n) \in \overline{K}^n$ une base de vecteurs propres communs à $S_1, \ldots, S_s$. Soit $\lambda_{i,j}$ la valeur propre de S_j associée à v_i,* i. e.*, $S_j\, v_i = \lambda_{i,j}\, v_i$. Alors $S\, v_i = (\sum_{j=1}^{s} t_j\, \lambda_{i,j}) v_i$ et, en particulier, S a toutes ses valeurs propres dans $\overline{K}[t_1, \ldots, t_s]$.*

Une conséquence intéressante de ce corollaire est que les valeurs propres de M peuvent se calculer directement sans calculer préalablement celles des M_i. Pour ce faire, il nous suffit de factoriser sur $\overline{K}[t_1, \ldots, t_s]$ le déterminant de M en produit de facteurs linéaires (par exemple avec l'algorithme décrit dans [HRUW98, Appendix]). En effet, nous savons que $\det(M)$ est égal à $(-1)^n$ multiplié par le produit des valeurs propres de M. Or, d'après le corollaire 2, ces valeurs propres sont des formes linéaires en les t_i à coefficients dans $\overline{K}$ donc $\det(M)$ se factorise nécessairement sur $\overline{K}[t_1, \ldots, t_s]$ en produit de formes linéaires.

L'algorithme IsoDecSim1 suivant calcule une décomposition isotypique simultanée de $\{M_1, \ldots, M_s\}$.

Procédure IsoDecSim1
Entrée : $\{M_1, \ldots, M_s\}$ (avec $M_i \in \mathbb{M}_n(K)$) s matrices qui commutent.
Sortie : $P \in \mathbb{M}_n(K)$ donnant une décomposition isotypique des M_i.
1 - Soit $M := t_1\, M_1 + \cdots + t_s\, M_s$.
2 - Calculer et factoriser sur $K(t_1, \ldots, t_s)$ le polynôme caractéristique de M :
soit $\chi(M) = F_1^{m_1} \cdots F_d^{m_d}$ avec les F_i irréductibles sur $K(t_1, \ldots, t_s)$ et premiers entre eux,
3 - Pour $i \in \{1, \ldots, d\}$, faire :
  3a - Calculer une base $e_i = (e_{i,1}, \ldots, e_{i,n_i})$ de $\ker(F_i^{m_i}(M))$.
(choisir les $e_{i,j}$ ne dépendants pas des t_i)
4 - Retourner la matrice P ayant les e_{ij} pour colonnes.

La procédure donnée ci-dessus calcule une décomposition isotypique simultanée sur K de toute famille de matrices à coefficients dans K qui commutent. Maintenant, dans

le cas où les M_i sont toutes diagonalisables, alors nous pouvons calculer une diagonalisation simultanée sur $\overline{K}$: d'après la discussion suivant le corollaire 2, il nous suffit de remplacer les étapes 2 et 3a de l'algorithme IsoDecSim1 ci-dessus par :

2 Factoriser $\det(M)$ en produit de facteurs linéaires :
$\det(M) = (-1)^n \prod_{i=1}^{n} (\alpha_{i,1}\, t_1 + \cdots + \alpha_{i,s}\, t_s)$ avec les $\alpha_{i,j}$ dans $\overline{K}$.
3a Calculer une base de vecteurs propres $e_{i,j}$ associés à la valeur propre $\alpha_{i,1}\, t_1 + \cdots + \alpha_{i,s}\, t_s$ (choisir un représentant ne dépendant pas des t_i).

Exemple 13. *Soit $K := \mathbb{F}_p(x_1, x_2, x_3)$ avec $p = 7$ et calculons une décomposition isotypique simultanée de $\{M_1, M_2, M_3\}$ où $M_1 = I_4$, et où M_2 et M_3 sont les matrices suivantes (on peut vérifier que ces matrices commutent entre elles deux à deux) :*

$$M_2 := \begin{bmatrix} 6\,{x_1}^7 + 2\,x_3 + 4\,{x_2}^7 & 4\,x_3 + {x_2}^7 & 5\,{x_1}^7 + 6\,x_3 + 5\,{x_2}^7 & 6\,{x_1}^7 + {x_2}^7 \\ 2\,x_3 + 2\,{x_2}^7 & 2\,{x_1}^7 + 4\,x_3 & {x_1}^7 + 6\,x_3 + 2\,{x_2}^7 & 6\,{x_1}^7 + {x_2}^7 \\ 3\,{x_1}^7 + 5\,x_3 + 2\,{x_2}^7 & 4\,{x_1}^7 + 3\,x_3 + 2\,{x_2}^7 & {x_1}^7 + x_3 + 4\,{x_2}^7 & {x_1}^7 + 6\,{x_2}^7 \\ 3\,{x_1}^7 + 6\,{x_2}^7 & 6\,{x_1}^7 + 2\,{x_2}^7 & 2\,{x_1}^7 + {x_2}^7 & {x_2}^7 \end{bmatrix},$$

$$M_3 := \begin{bmatrix} 6\,{x_1}^7\,{x_2}^7 + 2\,{x_3}^{14} & 0 & 5\,{x_1}^7\,{x_2}^7 + 2\,{x_3}^{14} & {x_3}^{14} + 6\,{x_1}^7\,{x_2}^7 \\ 0 & 2\,{x_1}^7\,{x_2}^7 + 6\,{x_3}^{14} & {x_1}^7\,{x_2}^7 + 6\,{x_3}^{14} & {x_3}^{14} + 6\,{x_1}^7\,{x_2}^7 \\ 3\,{x_1}^7\,{x_2}^7 + 4\,{x_3}^{14} & 4\,{x_1}^7\,{x_2}^7 + 3\,{x_3}^{14} & {x_1}^7\,{x_2}^7 & {x_1}^7\,{x_2}^7 + 6\,{x_3}^{14} \\ 3\,{x_1}^7\,{x_2}^7 + 4\,{x_3}^{14} & {x_3}^{14} + 6\,{x_1}^7\,{x_2}^7 & 2\,{x_1}^7\,{x_2}^7 + 5\,{x_3}^{14} & {x_3}^{14} \end{bmatrix}.$$

Considérons la matrice

$$M := t_1\, M_1 + t_2\, M_2 + t_3\, M_3,$$

à coefficients dans $\mathbb{K}[t_1, t_2, t_3]$. Calculons et factorisons son polynôme caractéristique :

```
> F:=Factors(numer(charpoly(M,T))) mod p;
```

$$F := [1,\ [[T + 6\,t_2\,{x_2}^7 + 6\,t_3\,{x_3}^{14} + 6\,t_1,\ 2],\ [6\,t_3\,{x_1}^7\,{x_2}^7 + 6\,t_2\,{x_1}^7 + T + 6\,t_1,\ 2]]]$$

Nous avons donc deux facteurs premiers entre eux ce qui signifie que les trois matrices M_1, M_2 et M_3 vont être simultanément décomposables en deux blocs diagonaux. Calculons cette décomposition : en calculant une base (ne dépendant pas de t_1, t_2 et t_3) de chaque sous-espace caractéristique $N1$ et $N2$, nous obtenons la matrice de passage

```
> P:=concat(N1[1],N1[2],N2[1],N2[2]);
```

$$P := \begin{bmatrix} 6 & 5 & 6 & 3 \\ 4 & 3 & 6 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

qui donne la décomposition suivante de M :

```
> IP:=Inverse(P) mod p: M1:=map(X->Normal(X) mod p,evalm(IP&*M&*P));
```

$$M1 := \begin{bmatrix} t_1 + t_3\, x_3{}^{14} + 3\, t_2\, x_2{}^7 & 2\, t_2\, x_2{}^7 & 0 & 0 \\ 5\, t_2\, x_2{}^7 & t_1 + t_3\, x_3{}^{14} + 6\, t_2\, x_2{}^7 & 0 & 0 \\ 0 & 0 & t_1 + t_3\, x_1{}^7\, x_2{}^7 + t_2\, x_1{}^7 & 4\, t_2\, x_3 \\ 0 & 0 & 0 & t_1 + t_3\, x_1{}^7\, x_2{}^7 + t_2\, x_1{}^7 \end{bmatrix}$$

En substituant les valeurs de t_1, t_2 et t_3 nécessaires (on aurait aussi pu calculer les $P^{-1}\, M_i\, P$), on trouve :

```
> map(X->Normal(X) mod p,subs({t_1=1,t_2=0,t_3=0},M1));
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
> map(X->Normal(X) mod p,subs({t_1=0,t_2=1,t_3=0},A1));
```

$$\begin{bmatrix} 3\, x_2{}^7 & 2\, x_2{}^7 & 0 & 0 \\ 5\, x_2{}^7 & 6\, x_2{}^7 & 0 & 0 \\ 0 & 0 & x_1{}^7 & 4\, x_3 \\ 0 & 0 & 0 & x_1{}^7 \end{bmatrix}$$

```
> map(X->Normal(X) mod p,subs({t_1=0,t_2=0,t_3=1},A1));
```

$$\begin{bmatrix} x_3{}^{14} & 0 & 0 & 0 \\ 0 & x_3{}^{14} & 0 & 0 \\ 0 & 0 & x_1{}^7\, x_2{}^7 & 0 \\ 0 & 0 & 0 & x_1{}^7\, x_2{}^7 \end{bmatrix}$$

Nous avons montré précédemment comment diagonaliser simultanément (calculer une décomposition isotypique simultanée) une famille de matrices qui commutent entre elles deux à deux. Supposons maintenant que nos s matrices $M_1, \ldots, M_s$ ne sont pas

simultanément diagonalisables par blocs et voyons comment les trianguläriser simultanément. Considérons une nouvelle fois la matrice

$$M = t_1\, M_1 + \cdots + t_s\, M_s,$$

où les t_i sont des indéterminées. Alors, en raisonnant comme nous l'avons fait précédemment, il n'est pas difficile de voir qu'en triangularisant cette matrice M (par exemple en suivant la méthode usuelle - voir [CO68, Chapitre 19, Section 7]) sur $\overline{K}[t_1, \ldots, t_s]$, nous obtenons en particulier la triangularisation simultanée sur $\overline{K}$ des M_i. De la même façon, si nous voulons triangulariser simultanément par blocs les M_i sur K, alors il nous suffit de triangulariser par blocs sur K la matrice M.

Exemple 14. *Nous nous proposons de triangulariser simultanément sur* $\mathbb{Q}$ *les matrices* M_1 *et* M_2 *suivantes (*M_1 *et* M_2 *commutent) :*

$$M_1 := \begin{bmatrix} \frac{9373}{1371} & \frac{-961}{1371} \\ \frac{1369}{1371} & \frac{7079}{1371} \end{bmatrix},$$

$$M_2 := \begin{bmatrix} \frac{9373}{8226} & \frac{-961}{8226} \\ \frac{1369}{8226} & \frac{7079}{8226} \end{bmatrix}.$$

Considérons la matrice $M = t_1\, M_1 + t_2\, M_2$ *à coefficients dans* $\mathbb{Q}[t_1, t_2]$ *et factorisons son polynôme caractéristique.*

```
> M:=evalm(t_1*M_1+t_2*M_2);
```

$$M := \begin{bmatrix} \frac{9373\, t_1}{1371} + \frac{9373\, t_2}{8226} & -\frac{961\, t_1}{1371} - \frac{961\, t_2}{8226} \\ \frac{1369\, t_1}{1371} + \frac{1369\, t_2}{8226} & \frac{7079\, t_1}{1371} + \frac{7079\, t_2}{8226} \end{bmatrix}$$

```
> factors(charpoly(M,T));
```

$$[1, [[T - 6\, t_1 - t_2, 2]]]$$

Calculons le vecteur propre associé à l'unique valeur propre de M*.*

```
> N_1:=nullspace(evalm(subs(T=G,T-6*t_1-t_2)))[1];
```

$$N_1 := \left[1, \frac{37}{31}\right]$$

L'espace propre associé est de dimension 1 *et donc* M *n'est pas diagonalisable. Pour trianguláriser* M*, il nous suffit de compléter* N_1 *en une base de* $\mathbb{Q}^2$*. Ici le vecteur* $[1, 0]$ *convient et nous obtenons ainsi la matrice de changement de base suivante :*

```
> P:=concat(N_1,[1,0]);IP:=inverse(P):
```

$$P := \begin{bmatrix} 1 & 1 \\ \frac{37}{31} & 0 \end{bmatrix}$$

```
> evalm(IP&*A&*P);
```

$$\begin{bmatrix} 6\,t_1 + t_2 & \frac{1147\,t_1}{1371} + \frac{1147\,t_2}{8226} \\ 0 & 6\,t_1 + t_2 \end{bmatrix}$$

Nous avons alors triangularisé simultanément M_1 *et* M_2 *:*

```
> evalm(IP&*M_1&*P);
```

$$\begin{bmatrix} 6 & \frac{1147}{1371} \\ 0 & 6 \end{bmatrix}$$

```
> evalm(IP&*M_2&*P);
```

$$\begin{bmatrix} 1 & \frac{1147}{8226} \\ 0 & 1 \end{bmatrix}$$

Dans ce qui précède, nous avons vu comment diagonaliser et trianguláriser simultanément des matrices qui commutent. La réduction simultanée de matrices qui commutent à d'autres formes normales (*e.g.*, Jordan, Frobenius, ...) est plus compliquée notamment du fait que ces formes normales de matrices ne sont pas stables par addition (par exemple la somme de deux matrices sous forme de Jordan n'est en général pas une matrice sous forme de Jordan). Précisons encore que des matrices qui commutent ne sont pas nécessairement simultanément Jordanisables : un exemple explicite est donné dans [Ca94, 3.1].

Conclusion de la section 1.8

Pour généraliser l'algorithme développé dans les sections 1.3 à 1.6 au cas des systèmes d'équations aux dérivées partielles D-finis en caractéristique p, nous avons défini des notions de p-courbure et d'eigenring et nous avons exhibé les liens qui existent entre ces deux notions. Dans ce nouveau cadre, nous ne disposons plus d'une classification des

modules différentiels en caractéristique p ; cependant le fait que, dans le cas différentiel ordinaire, nous avions énoncé et prouvé certains résultats, découlant directement de la classification, d'une manière élémentaire (et algorithmique), nous a permis de généraliser ici ces résultats sans problèmes. En utilisant l'hypothèse de D-finitude, qui implique immédiatement que les p-courbures partielles commutent entre elles deux à deux, nous avons montré qu'en suivant l'approche faite dans le cas différentiel ordinaire, notre algorithme de factorisation se ramène au problème de la réduction simultanée de matrices qui commutent.

En nous concentrant sur ce dernier problème, nous avons montré comment diagonaliser (resp. triangulariser) simultanément des matrices qui commutent. Pour ce faire, nous avons considéré une combinaison linéaire à coefficients indéterminés de nos matrices : en effet, en diagonalisant (resp. triangularisant) cette combinaison linéaire, nous diagonalisons (resp. triangularisons) en particulier chaque matrice. Les formes diagonales et triangulaires sont les deux formes normales qui nous servent à factoriser les systèmes différentiels en caractéristique p (voir les sections 1.4 et 1.5 de ce chapitre) : par conséquent, nous avons développé tous les ingrédients nécessaires pour obtenir un algorithme de factorisation de systèmes d'équations aux dérivées partielles D-finis en caractéristique p.

En ce qui concerne la réduction simultanée à d'autres formes normales (Frobenius, Jordan, ...), il semble que nous puissions aussi obtenir des informations intéressantes en regardant une combinaison linéaire à coefficients indéterminés et nous continuons à travailler sur ce problème.

Chapitre 2

A Modular Algorithm for Computing the Exponential Solutions of a Linear Differential Operator

Ce chapitre constitue une publication ([CH04]) co-écrite avec Mark van Hoeij et parue dans *Journal of Symbolic Computation* : le texte a été laissé dans sa langue d'origine.

Introduction

An ordinary differential equation

$$y^{(n)} + a_{n-1}y^{(n-1)} + \cdots + a_0 y = 0 \tag{2.1}$$

corresponds to a differential operator

$$\partial^n + a_{n-1}\partial^{n-1} + \cdots + a_0\partial^0 \tag{2.2}$$

acting on y. Let $C \subset \overline{\mathbb{Q}}$ be some number field (we will allow more general C in Section 2.7.1). The topic of this paper is finding *exponential solutions* of (2.1) when the a_i are in $C(x)$. By this we mean : finding all $r \in \overline{\mathbb{Q}}(x)$ for which $y = \exp(\int r)$ is a solution of (2.1). This is equivalent to finding *first order right hand factors* $\partial - r$ with $r \in \overline{\mathbb{Q}}(x)$ of (3.1). Consequently, it can be viewed as part of the more general problem of factoring differential operators. Beke gave two factoring algorithms in [Be1894] (see also [PS03, 4.1 and 4.2.1]), one algorithm for first order factors (which we will refer to as *Beke's algorithm* in this paper) and one algorithm (which uses first order factors) for higher order factors. Other factoring algorithms are the local to global method of [Ho97a, Ho97b] and the eigenring method ([Si96], [PS03, 4.2.2]). Computing exponential solutions (computing first order factors) with these algorithms can be slow. It is not difficult to find examples where Maple's *expsols* takes a long time, even though it uses

an almost rational version of Beke's algorithm, the local to global method, as well as the eigenring method. The goal in this paper is to give a faster algorithm. We combine modular methods with an almost rational version of Beke's algorithm to obtain Algorithm ExpSolsInC which computes the exponential solutions defined over a given field C. Then we study the field problem, which is necessary for giving an efficient algorithm ExpSols for computing all exponential solutions.

We sketch Beke's method using the vocabulary of this paper. First, at each *singularity*, one computes a finite number (at most n) of objects called *generalized exponents*. Then, for each *combination* of generalized exponents (choose one generalized exponent at each singularity), the problem reduces to computing polynomial solutions (*e.g.*, with [ABP95]) of some other differential operator. There are two main problems :

- *a combinatorial problem* : one has to choose a generalized exponent at each singularity which could lead to a large number (at most n^m where m is the number of singularities) of possible combinations.
- *a field problem* : singularities and generalized exponents are often defined over algebraic extensions of C, so some combinations can be defined over large algebraic extensions of C.

If neither problem occurs then Beke's algorithm usually works well in practice. Thus we are mainly interested in situations where one or both of these two problems do occur. If we can discard combinations in advance then that would address the combinatorial problem, and in some cases even the field problem if the discarded combinations include those that are defined over large extensions of C. Our first main goal is to do this with computations modulo a prime number p.

The algebraic theory of ordinary differential equations in characteristic p is studied in [Ho81, Ka82, CC85, Pu95]. For the problem of factoring differential operators with coefficients in $\overline{\mathbb{F}}_p(x)$, van der Put showed in [Pu95] (this is also recalled in [Pu96, 2.1] and [PS03, 13.1]) that the relevant object is the *p-curvature* since it determines the differential module mod p and all factorizations of the operator. More recently, an algorithm to factor Ore polynomials with coefficients in $\mathbb{F}_p(x)$ is included in [GZ03]. This algorithm is based on the eigenring method adapted to characteristic p and generalizes the work of [Gi98] that tackles the problem of factoring differential operators over $\mathbb{F}_q$ where $q = p^l$ for p prime and $l \in \mathbb{N}^*$. An algorithm to factor differential systems over $\overline{\mathbb{F}}_p(x)$, giving links between the p-curvature and the eigenring, is found in [Cl03].

The natural idea to lift modular factorizations to characteristic zero has been studied in [Pu96, 4.3] and [PS03, 13.2.3]. In this paper we propose a different approach because there are many problems that prevent lifting from becoming an efficient and complete algorithm for characteristic zero. For example, there could be many factorizations in characteristic p, and it is not clear which of these should be lifted. Also, it is not clear how to do efficient Hensel lifting. Using more than one prime leads to the combinatorial problem of combining factors of different characteristics, and using large prime(s) makes p-curvature computations or factoring operators mod p expensive. We propose not to use factors of the operator mod p for problems in characteristic zero, but to use only the characteristic polynomial of the p-curvature.

To reduce the number of combinations to check in Beke's algorithm, we prove links

between eigenvalues in $\overline{\mathbb{F}}_p(x^p)$ of the p-curvature and the reduction mod p of the generalized exponents appearing in possible solutions. Algorithm CombMatchRoot determines which generalized exponents match a given eigenvalue. Algorithm ExpSolsInC uses this information to compute all exponential solutions defined over a field C (this C must be given in the input). Unfortunately, there are cases (see Section 5) where our mod p computations yield no improvements for the combinatorial problem.

The *field problem* refers to the problem that the usual Beke's algorithm computes with a generalized exponent at *every singularity simultaneously*. So the degree of the field extension of C in Beke's algorithm depends *exponentially* on m, the number of singularities (an extension of degree $\leq m!$ for the singularities followed by an extension of degree $\leq n^m$ for the generalized exponents). Of course this can easily choke the computation. Algorithm ExpSolsInC does not have this large field problem because the highest degree extensions of C it uses are those given by *just one singularity*. In fact, on complicated examples usually most of the work in ExpSolsInC consists of rational (*i.e.*, over C) computations and not of computations in field extensions. However, for ExpSolsInC to find an exponential solution y, it needs in its input a field over which y can be defined. From now on, the *field problem* refers to the problem of finding such a field without constructing very large extensions that choke the computation. Our second main goal is solving this field problem.

Our final algorithm ExpSols first calls ExpSolsInC, then uses modular information to check if more exponential solutions could exist, and if so, it will first reduce the order as much as possible before tackling the field problem. After that, it calls the recursive FindASol algorithm which solves the field problem and returns an exponential solution defined over an extension, if such solution exists. We give several bounds (one of which is based on the p-curvature) for the degree of a field extension needed to find an exponential solution. Algorithm FindASol uses these bounds and makes an algebraic extension to decrease these bounds. It does this recursively and each time it calls ExpSolsInC to search for a solution over the new field. This continues until either a solution is found or until the bound becomes 1.

We summarize the two main new results in this paper :

- We use information mod p in several ways (in Section 5 for the combinatorial problem and Section 6 for improving bounds that are used in the field problem) to speed up computation of exponential solutions in characteristic 0.
  To accomplish this there is a subtle but important difficulty to be overcome. There are many results known that hold for almost all p, but if such p can not easily be identified (an example is given in Section 2.7.3) then the algorithm can not use the result. *The algorithm needs to know which p it can use*. This restriction has numerous consequences, and the algorithm needs to be designed with this issue in mind. We will give a notion of good primes in Definition 46, and then only use results that hold for all good p. Another important issue is that the definition of good primes should discard as few primes as possible (for large p the p-curvature computation becomes slow, so only small p are useful for speeding up ExpSols).
- Our second main new result is Algorithm FindASol in Section 2.6.2, a recursive algorithm that constructs a field extension over which an exponential solution (if

one exists) can be defined.

An efficiency advantage of Beke's method over the local to global method in [Ho97b] or the eigenring method in [Si96] is that it directly computes first order factors. In contrast, the local to global method often needs to compute higher order factors before it can reach the first order factors, and the eigenring method first has to compute the eigenring. The greatest disadvantage of Beke's method in comparison to the eigenring and local to global approaches is its use of a splitting field, and this is the issue that Algorithm FindASol resolves.

The paper is organized as follows. Section 1 contains the vocabulary needed to read this paper, in particular the notions of generalized exponents, reduction mod p, and p-curvature. Section 2 illustrates in a few examples how mod p computations can reduce the combinatorial problem. In Sections 3 and 4 we give links between the p-curvature and generalized exponents. Section 5 contains Algorithm CombMatchMod$p$ that reduces the number of combinations to check and Algorithm ExpSolsInC that computes all exponential solutions defined over a given field C. Section 6 addresses the field problem. Combining these algorithms in Section 7 we give Algorithm ExpSols that computes a basis of all exponential solutions, up to conjugation over C.

2.1 Preliminaries

In this section we briefly recall the concepts that are needed to read the next sections. The first subsection contains the necessary material in characteristic zero, the second subsection deals with characteristic p, and the third deals with reduction from characteristic 0 to characteristic p.

2.1.1 Differential operators in characteristic zero

Let $\overline{\mathbb{Q}}$ denote the algebraic closure of $\mathbb{Q}$. We endow the fraction field $\overline{\mathbb{Q}}(x)$ of the ring $\overline{\mathbb{Q}}[x]$ of polynomials in the indeterminate x with the usual differentiation $' := \frac{d}{dx}$. The *field of constants* of *the differential field* $(\overline{\mathbb{Q}}(x),')$ is then $\overline{\mathbb{Q}}$. Now, we consider the non-commutative ring $\overline{\mathbb{Q}}(x)[\partial]$ of differential operators over the differential field $(\overline{\mathbb{Q}}(x),')$; in this ring, the multiplication is given by $\forall\, u \in \overline{\mathbb{Q}}(x)$, $\partial u = u\partial + u'$. After multiplying if necessary by some element of $\overline{\mathbb{Q}}(x)$, a non-zero element $L \in \overline{\mathbb{Q}}(x)[\partial]$ can be written as

$$L = a_n\partial^n + a_{n-1}\partial^{n-1} + \ldots + a_0\partial^0,\ a_i \in \overline{\mathbb{Q}}[x],\ a_n \neq 0 \text{ and } \gcd(a_0, \ldots, a_n) = 1. \quad (2.3)$$

As usual, n will be called *the order* of the differential operator L.

Singularities and generalized exponents

We start with some classical definitions that can be found for example in [In26].

Definition 1. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$ be written as in (2.3). A point $x_0 \in \overline{\mathbb{Q}}$ is said to be* a singular point (or a singularity) of L *if it is a root of a_n. Otherwise, it is said to be* a regular (or ordinary) point of L.

In this paper we always consider the point at infinity as a singular point, whether it is singular or not. To each differential operator $L \in \overline{\mathbb{Q}}(x)[\partial]$, we associate the subset $\mathcal{S}$ of $\overline{\mathbb{Q}}$ containing all finite singularities. The set of all singularities of L is then $\overline{\mathcal{S}} := \mathcal{S} \cup \{\infty\}$.

Let $\delta := x\partial$ and remark that $\overline{\mathbb{Q}}(x)[\partial] = \overline{\mathbb{Q}}(x)[\delta] \subset \overline{\mathbb{Q}}((x))[\delta]$.

Definition 2. *Let L be a non-zero differential operator in $\overline{\mathbb{Q}}((x))[\delta]$. L can be written $\sum_{i=v}^{\infty} x^i p_i(\delta)$ for some polynomials p_i over $\overline{\mathbb{Q}}$ and $p_v \neq 0$. Then* the indicial equation of *L is the polynomial p_v. Its roots in $\overline{\mathbb{Q}}$ are called* the exponents of L at $x = 0$ *(see [In26, 7.21] or [HW97, Lemma 11]).*

The notion of *exponents* can be generalized in the sense of [HW97, 3.2] : there are several distinct ways to define *generalized exponents*. We choose here the algebraic definition of [Ho97a, Ho97b] that is easy to extend to characteristic p. We first give the definition at the point $x = 0$ and then extend it to all points. At a point $x_i \in \overline{\mathbb{Q}} \cup \{\infty\}$, we define the *local parameter* t_i by

$$t_i := \begin{cases} x - x_i & \text{if} \quad x_i \in \overline{\mathbb{Q}}, \\ 1/x & \text{if} \quad x_i = \infty. \end{cases}$$

Definition 3. *Let $e \in \overline{\mathbb{Q}}[x^{-\frac{1}{k}}]$ with k minimal in $\mathbb{N}^*$ and let $L_{\delta \to \delta+e}$ denote the differential operator $L \in \overline{\mathbb{Q}}(x)[\delta]$ in which we have replaced δ by $\delta + e$. Then e is said to be* a generalized exponent of L at $x = 0$ *if 0 is a root of the indicial equation of $L_{\delta \to \delta+e}$. The number k is called* the ramification (index) of e. *The multiplicity of the root 0 in the indicial equation is called* the multiplicity of e. *If $k = 1$ then e is called an* unramified generalized exponent. *If $k = 1$ and L has no generalized exponent $\tilde{e}$ for which $\tilde{e} - e$ is a negative integer then e is called a* $\mathbb{Z}$-minimal generalized exponent.

Definition 4. *Let $e \in \overline{\mathbb{Q}}[t_i^{-\frac{1}{k}}]$. Then e is said to be* a generalized exponent of L at $x = x_i$ *if $e_{t_i \to x}$ (e in which we have replaced t_i by x) is a generalized exponent of $L_{t_i \to x}$ at $x = 0$. Here $L_{t_i \to x}$ refers to the automorphism of $\overline{\mathbb{Q}}(x)[\partial]$ given by $x \mapsto x + x_i$ at a finite point and $x \mapsto 1/x$ at infinity. This leads to $\partial \mapsto \partial$ at a finite point and $\partial \mapsto -x^2\partial$ at infinity.*

Generalized exponents are useful for computing exponential solutions, see Lemma 1 and Remark 3. At each point x_i, there are exactly n generalized exponents counted with multiplicity. The point x_i is called a *regular singular* point of L when all generalized exponents are constants, in which case they are just the usual exponents. At a regular point x_i the (generalized) exponents are $0, 1, \ldots, n-1$. For further details on generalized exponents see [Ho97a, Ho97b, HW97].

Exponential solutions and first order right hand factors

Definition 5. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$ and let $y \neq 0$ belong to a differential field extension of $\overline{\mathbb{Q}}(x)$ with $\overline{\mathbb{Q}}$ as field of constants. Assume that $L(y) = 0$.*
Then, y is said to be an exponential solution of L *if $y'/y \in \overline{\mathbb{Q}}(x)$. It is said to be* a radical solution *if there exists a positive integer m such that $y^m \in \overline{\mathbb{Q}}(x)$ and finally, it is said to be* a rational *(respectively* polynomial*)* solution *if $y \in \overline{\mathbb{Q}}(x)$ (respectively $y \in \overline{\mathbb{Q}}[x]$).*

An exponential solution y of L corresponds to the first order right hand factor $\partial - y'/y$ of L.

Remark 1. *In this paper it will not be necessary to distinguish an exponential function y from a scalar multiple of y. Hence we may represent y by its logarithmic derivative $y'/y \in \overline{\mathbb{Q}}(x)$ or by its minimal annihilating operator $\partial - y'/y$.*

Definition 6. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$ be of order $n > 1$. $R \in \overline{\mathbb{Q}}(x)[\partial]$ of order $0 < m < n$ is said to be a* right hand factor of L *if there exists $l \in \overline{\mathbb{Q}}(x)[\partial]$ such that $L = lR$.*

Lemma 1 below (analogous to [Bou00, Proposition 8]) shows that the notions of generalized exponent, exponential solution and first order right hand factor are closely related.

Lemma 1. *Let $r \in \overline{\mathbb{Q}}(x)$. If $\partial - r$ is a first order right hand factor of $L \in \overline{\mathbb{Q}}(x)[\partial]$, then for all x_i in $\overline{\mathcal{S}}$, there exists a generalized exponent $e_{x_i} \in \overline{\mathbb{Q}}[t_i^{-1}]$ of L at x_i such that*

$$r = S + \frac{Q'}{Q} \quad \text{where} \quad S = \sum_{x_i \in \mathcal{S}} \frac{e_{x_i}}{t_i} - t_\infty e_\infty^*,$$

and (Fuchs' relation)

$$\deg(Q) + \sum_{x_i \in \overline{\mathcal{S}}} \mathrm{Const}(e_{x_i}) = 0, \quad \text{in particular} \quad -\sum_{x_i \in \overline{\mathcal{S}}} \mathrm{Const}(e_{x_i}) \in \mathbb{N} \tag{2.4}$$

where $Q \in \overline{\mathbb{Q}}[x]$, the $\mathrm{Const}(e_{x_i})$ are the constant terms of the e_{x_i} and $e_\infty^* := e_\infty - \mathrm{Const}(e_\infty)$.

Proof. Let e_{x_i} be the generalized exponent of $\partial - r$ at x_i. Since $\partial - r$ is a right hand factor of L, this is also a generalized exponent of L (see [Ho97b, Lemma 3.2]). Now $L_{\partial \to \partial + S}$ has right hand factor $\partial - \tilde{r}$ where $\tilde{r} = r - S$. Let $\mathcal{S}'$ be the set of all finite singularities of $\partial - r$ and let $\Delta = \mathcal{S}' \setminus \mathcal{S}$. So $x_i \in \Delta$ if and only if x_i is a regular point of L (hence $e_{x_i} \in \mathbb{N}$) but a singular point (hence $e_{x_i} \neq 0$) of $\partial - r$. Now $r = \sum_{x_i} \frac{e_{x_i}}{t_i} - t_\infty e_\infty^*$ if we take the sum over all finite singularities of $\partial - r$. Hence $\tilde{r} = \sum_{x_i \in \Delta} \frac{e_{x_i}}{t_i}$. Then $\partial - \tilde{r}$ has a polynomial solution $Q = \prod_{x_i \in \Delta} t_i^{e_{x_i}}$. Fuchs' relation (which applied to $\partial - r$ is just a reformulation of the fact that the sum of the residues of r is zero) says that if we sum $\mathrm{Const}(e_{x_i})$ over all singularities of $\partial - r$ then the result is zero, $\sum_{x_i \in \overline{\mathcal{S}} \cup \Delta} \mathrm{Const}(e_{x_i}) = 0$. Then Equation (2.4) follows from the observation that $\deg(Q) = \sum_{x_i \in \Delta} e_{x_i}$.□

Remark 2. *In our algorithm we only need to compute* unramified *generalized exponents of L, those are the only ones that can be* relevant *for our algorithm.*

If at a singularity $x_i \in \mathcal{S}$ there are two generalized exponents $e_{x_i,1}$ and $e_{x_i,2}$ whose difference $d = e_{x_i,1} - e_{x_i,2}$ is a positive integer, then $e_{x_i,1}$ is not relevant *in the sense that the lemma stays true if we never use $e_{x_i,1}$ (because replacing $e_{x_i,1}$ and Q by $e_{x_i,2}$ and $t_i^d Q$ does not change $S + Q'/Q$). So, in our algorithm, we will only use* $\mathbb{Z}$-minimal *generalized exponents.*

The r defined in the previous lemma matches the exponential solution

$$\exp(\int S)\, Q.$$

If all e_{x_i} are rational numbers then this exponential solution can be written as

$$Q \prod_{x_i \in \mathcal{S}} t_i^{e_{x_i}} \tag{2.5}$$

so this is a radical solution. We see then that the radical solutions correspond to the exponential solutions for which all associated generalized exponents are rational numbers.

Remark 3. *If we only know the generalized exponents defining some exponential solution modulo rational numbers (respectively modulo integers), then finding such exponential solution reduces to finding radical solutions (respectively rational solutions) of* $L_{\partial \to \partial + S}$.

2.1.2 Differential operators in characteristic p

Let $\overline{\mathbb{F}}_p$ denote the algebraic closure of the finite field $\mathbb{F}_p$. We define the ring $\overline{\mathbb{F}}_p(x)[\partial]$ of differential operators in the same way as in characteristic zero. The main difference is that the field of constants of $(\overline{\mathbb{F}}_p(x), ')$ is $\overline{\mathbb{F}}_p(x^p)$ (see [Pu01] or [GZ03, Lemma 3.3]).

The p-curvature and the map $y \mapsto y^{(p-1)} + y^p$

The central tool for studying differential operators in characteristic p is the p-curvature; we recall a definition and we refer to [Ka82, CC85] and [PS03, 13.2.2] for more information. The differential field $\overline{\mathbb{F}}_p(x)$ is a finite dimensional vector space over its field of constants $\overline{\mathbb{F}}_p(x^p)$:

$$\overline{\mathbb{F}}_p(x) = \bigoplus_{i=0}^{p-1} \overline{\mathbb{F}}_p(x^p) x^i.$$

To any differential operator $L \in \overline{\mathbb{F}}_p(x)[\partial]$, one can associate the differential module

$$\mathcal{M}_L := \overline{\mathbb{F}}_p(x)[\partial] / \overline{\mathbb{F}}_p(x)[\partial] L.$$

This module is equipped with a $\overline{\mathbb{F}}_p(x^p)$-linear map $\tilde{\partial}$ (coming from the left multiplication by ∂ on $\overline{\mathbb{F}}_p(x)[\partial]$) satisfying

$$\tilde{\partial} f m = f' m + f \tilde{\partial} m \quad \text{for all} \quad m \in \mathcal{M}_L,\ f \in \overline{\mathbb{F}}_p(x).$$

Definition 7. *Let $L \in \overline{\mathbb{F}}_p(x)[\partial]$. The* p-curvature of L *is the $\overline{\mathbb{F}}_p(x)$-linear map $\tilde{\partial}^p$ acting on the differential module $\mathcal{M}_L$ associated to L.*

An algorithm to compute the p-curvature matrix is given in [PS03, 13.2.2]. The map

$$\begin{array}{rccc} \tau: & \overline{\mathbb{F}}_p(x) & \to & \overline{\mathbb{F}}_p(x^p), \\ & y & \mapsto & y^{(p-1)} + y^p \end{array}$$

is very useful when we deal with order one right hand factors in characteristic p. Details can be found in [[PS03], 13.2.1] and [Pu95] ; we only recall a lemma that we will often use :

Lemma 2 ([Pu95], Lemma 1.4.2). *The map $\tau : \overline{\mathbb{F}}_p(x) \to \overline{\mathbb{F}}_p(x^p)$ is a surjective additive map with kernel $\{ \frac{z'}{z} \mid z \in \overline{\mathbb{F}}_p(x)^* \}$.*

In [Pu95], van der Put proves that for the problem of factoring a differential operator in characteristic p (or in an equivalent way its associated differential module $\mathcal{M}_L$), the p-curvature is the relevant object since its characteristic properties give all possible factorizations of the operator.

We will note $\chi_p(L)$ the characteristic polynomial of the p-curvature of $L \in \overline{\mathbb{F}}_p(x)[\partial]$. As we are only interested in first order factors, the only modular information that we will use are the roots in $\overline{\mathbb{F}}_p(x^p)$ (with multiplicities) of $\chi_p(L)$. We denote $\mathcal{R}(\chi_p(L))$ as the set of roots of $\chi_p(L)$ in $\overline{\mathbb{F}}_p(x^p)$.

Lemma 3. *Let $L \in \overline{\mathbb{F}}_p(x)[\partial]$.*
(a). *If $L = L_1 L_2$ then $\chi_p(L) = \chi_p(L_1)\chi_p(L_2)$ with $\deg(\chi_p(L_i)) = order(L_i)$.*
(b). *The map μ from the set of all first order monic right hand factors of L in $\overline{\mathbb{F}}_p(x)[\partial]$ to $\mathcal{R}(\chi_p(L))$ given by $\mu(\partial - r) := \tau(r)$ is well defined and surjective.*

Proof. The claim (a) follows from the classification (the equivalence of categories) of [Pu95] and [PS03, 13.1]. That the map μ is well defined follows from the first claim and the fact that $\tau(r)$ is the p-curvature of the order one operator $\partial - r$ (see [Pu96, Lemma 2.2]). Surjectivity follows from the classification as well.□

2.1.3 Reduction modulo p and factorization

Let $v_p : \mathbb{Q} \to \mathbb{Z} \cup \{\infty\}$ denote the standard p-adic valuation on $\mathbb{Q}$. There are infinitely many ways to extend this valuation to a valuation $v_p : \overline{\mathbb{Q}} \to \mathbb{Q} \cup \{\infty\}$. We assume that one such choice is made. Let $R_p = \{a \in \overline{\mathbb{Q}} \mid v_p(a) \geq 0\}$, and consider the maximal ideal $I = \{a \in R_p \mid v_p(a) > 0\}$. Now choose an isomorphism $R_p/I \cong \overline{\mathbb{F}}_p$.

Definition 8. *Having fixed the above choices, we denote the image of $a \in R_p$ in $\overline{\mathbb{F}}_p$ by $a[p]$. This is called the* reduction *of a mod p.*

Note that $a[p]$, the reduction of a mod p, depends not only on a and p but depends on the choice of v_p as well. However, for compactness of notation we will only mention p even when we refer to both p and v_p. If $a \in \overline{\mathbb{Q}}$ is not in R_p then $a[p]$ is not defined. In our algorithm, we will apply the reduction map $[p] : R_p \to \overline{\mathbb{F}}_p$ to only finitely many algebraic numbers that are known in advance. Thus, it is not difficult to choose a prime p and valuation v_p in such a way that $a[p]$ is always defined for all numbers a for which

the reduction map $[p]$ is used. The a's we reduce mod p are finite in number and thus they are elements of a finite extension K of $\mathbb{Q}$. There are only finitely many possible choices for v_p on K. We will assume that such p and v_p are chosen, so that all instances of the notation $a[p]$ are defined. For more details on this choice see Section 5.3.

In practice, reducing an algebraic number a mod p is done as follows : Let $a_1, \ldots, a_n$ be those algebraic numbers that have previously been reduced mod p, but take only those whose reduction mod p may appear in the same expression as $a[p]$ in the algorithm (if the reductions of $a_1, \ldots, a_n$ and some other numbers $b_1, \ldots, b_m$ are used completely independently, then the algorithm will still be valid if it computes the reductions of these two sets of numbers independently, using one valuation on $\mathbb{Q}(a_1, \ldots, a_n)$ and an independently chosen valuation on $\mathbb{Q}(b_1, \ldots, b_m)$. This way the larger field $\mathbb{Q}(a_1, \ldots, a_n, b_1, \ldots, b_m)$ does not need to be constructed). Then determine the minimal polynomial of a over $K := \mathbb{Q}(a_1, \ldots, a_n)$. Reduce this minimal polynomial mod p, then the coefficients are in a field we will denote by $K[p]$, which is defined as $\mathbb{F}_p(a_1[p], \ldots, a_n[p])$. Factor this polynomial over $K[p]$, and choose one of the irreducible factors. Then determine a field extension of $K[p]$ in which this irreducible factor has a root, and let $a[p]$ be that root.

We will apply the reduction map $[p]$ to the singularities of L, which are in $\overline{\mathbb{Q}} \cup \{\infty\}$. If $a = \infty$ then we define $a[p]$ as ∞. Two distinct singularities of L can have the same image under $[p]$, in which case we say that p is not a good prime (see condition $(C3)$ in Definition 46 in Section 3). Note that this definition of good prime depends not only on p, but depends on v_p as well (recall that when we refer to both p and v_p we will only mention p for brevity).

If a is some object defined over $\overline{\mathbb{Q}}$ (like L, a local parameter t_i, or a generalized exponent e_{x_i}) we can define $a[p]$ by applying the reduction map $[p]$ to all coefficients.

We show now that without too many assumptions on p a factorization of a differential operator in characteristic zero can be reduced mod p. This result can be found in [Pu96].

Proposition 1. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$ and let p be such that L can be reduced mod p. If $L = L_1 L_2$ then, after possibly replacing L_1 and L_2 by cL_1 and $\frac{1}{c}L_2$ for some constant $c \in \overline{\mathbb{Q}}$, we have $L[p] = L_1[p]\ L_2[p]$.*

We allow to replace L_1 and L_2 by cL_1 and $\frac{1}{c}L_2$ for some constant c to avoid pathologies of the form $L_1 = \frac{1}{p}\partial$ and $L_2 = p\partial$.

Proof. The only thing to prove is that cL_1 and $\frac{1}{c}L_2$ can be reduced mod p for some constant c. Any p-adic valuation v_p on $\overline{\mathbb{Q}}$ can be extended to $\overline{\mathbb{Q}}[x]$ (this statement is a form of Gauss' lemma. To define $v_p(f)$ for a non-zero polynomial one takes the smallest valuation of the coefficients). The valuation can then be extended to the fraction field $\overline{\mathbb{Q}}(x)$, and then be extended to $\overline{\mathbb{Q}}(x)[\partial]$ (again, take the minimal valuation of the coefficients). Then $L[p]$ is defined if and only if $v_p(L) \geq 0$. Let c be a constant with valuation $-v_p(L_1)$, so $v_p(cL_1) = 0$, hence cL_1 can be reduced mod p. Then $v_p(\frac{1}{c}L_2) = v_p(cL_1) + v_p(\frac{1}{c}L_2) = v_p(cL_1\frac{1}{c}L_2) = v_p(L_1L_2) = v_p(L) \geq 0$, so $\frac{1}{c}L_2$ can be reduced mod p as well.$\square$

One obtains the following two criteria (these can also be easily deduced from [Pu95]) :

Corollary 1. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$ and let p be a prime such that L can be reduced mod p. Assume further that the order n of L does not drop after the reduction.*
If $\chi_p(L)$ is irreducible over $\overline{\mathbb{F}}_p(x^p)$ then L is irreducible over $\overline{\mathbb{Q}}(x)$.
If $\chi_p(L)$ has no roots in $\overline{\mathbb{F}}_p(x^p)$ then L has no exponential solutions.

Proof. It follows directly from the previous proposition, [Cl03, Theorem 4.1-Step 2] and Lemma 3(b).□

As an example, let $L = (x^2+x+8)\partial^2 + (-x^8+x+6)\partial + 1$ and $p = 3$. Then $\chi_p(L)$ has no roots and this is a very fast way to show that L has no exponential solutions. Without modular methods this would have taken much longer because one of the degree bounds (one ends up searching for polynomial solutions of some other operators) turns out to be very high in this example.

In the corollary, the hypothesis that the order does not drop can not be omitted. For example, the operator $(p\partial + 1)\partial$ is reducible in characteristic 0 but its reduction $L[p] = \partial$ is irreducible.

2.2 Examples

We will illustrate our modular improvements for the combinatorial problem (see the introduction) with a few examples. These improvements rely on Proposition 1 and Lemma 3(b) in the following way. A first order factor in characteristic zero leads to a first order factor in characteristic p by Proposition 1, which in turn corresponds to a root of $\chi_p(L)$ by Lemma 3(b).

2.2.1 Example 1

Consider the following differential operator :

$$L = \partial^3 - \frac{2x^2-x+4}{2x^2}\partial^2 - \frac{3x^3-4x^2-3x-2}{2x^4}\partial + \frac{2x^3-3x-2}{2x^4}.$$

The singularities are 0 and ∞. Generalized exponents are 0, $\frac{5}{2}+\frac{1}{x}$, $2+\frac{1}{x}$ at 0 and $-t_\infty^{-1}$, $-t_\infty^{-\frac{1}{2}}$, $t_\infty^{-\frac{1}{2}}$ at ∞ (recall that $t_\infty = \frac{1}{x}$). The usual Beke's algorithm will have three combinations to check since we only have one unramified generalized exponent at ∞, and three at 0. We will use this example to illustrate our modular method, even though our method generally does not speed up trivial examples (L is trivial for Beke's algorithm because the number of combinations is very small and there is no field problem).

Let $p = 3$ and note $c = x^p$. The characteristic polynomial $\chi_p(L)$ of the p-curvature matrix is $(\lambda^2 + \lambda/c^2 + 2/c + 1/c^4)(\lambda + 2)$ and has only the root $\lambda = 1$. We will see in Section 2.5.2 that this root excludes the choices $\frac{5}{2}+\frac{1}{x}$ and $2+\frac{1}{x}$ at 0; consequently, there remains only one combination to check.

For each of the 3 combinations, Beke's algorithm will check Fuchs' relation (Equation (2.4) in Lemma 1). This check will be very fast because since there is no field problem in this example, we do not have to construct a complicated number field in order to do this check. One of the three combinations passes this check, and it yields a solution. In this example, the modular method would not provide a speedup because it discarded only two combinations (which otherwise would have been discarded anyway by Fuchs' relation). In the next example, our modular method discards more combinations, and moreover, the discarded combinations include all "hard" combinations, *i.e.*, those not invariant under the Galois group of $\overline{\mathbb{Q}}$ over $\mathbb{Q}$.

2.2.2 Example 2

Let

$$L = 9(x^3-2)^5\partial^3 + (x^3-2)(2x^{10} - 12x^7 + 108x^5 + 24x^4 - 216x^2 - 16x - 9)\partial \\ -2x(190x^6 - 274x^3 - 27x - 212).$$

There are three finite singularities; the roots of x^3-2, with generalized exponents 2, $\frac{1}{36}\frac{\alpha^2}{(x-\alpha)} - \frac{1}{18}\alpha$ and $-\frac{1}{36}\frac{\alpha^2}{(x-\alpha)} + 4 + \frac{1}{18}\alpha$ where $\alpha := \mathrm{RootOf}(x^3-2)$. At infinity, the generalized exponents are $0, -\frac{5}{3}, -\frac{4}{3}$. So the usual Beke's algorithm will have 81 combinations to try. Let $p = 5$ and look at the reduction mod p of the singularities and generalized exponents. First, we factor $x^3-2 = (x+2)(x^2+3x+4)$ mod p, so α reduces to 3 as well as to $\beta := \mathrm{RootOf}(x^2+3x+4)$ in characteristic p. After reduction mod p the situation is then as follows : we have the singularity 3 with generalized exponents 2, $\frac{4}{x+2} - 1$, $-\frac{4}{x+2}$ and the singularities β with generalized exponents 2, $\frac{\beta^2}{(x-\beta)} + 3\beta$, $-\frac{\beta^2}{(x-\beta)} + 4 + 2\beta$. At infinity the generalized exponents become 0 (with multiplicity 2) and 2. Now, we compute $\chi_p(L)$ and see that its only root is $\lambda = s$ where $s := \frac{4c^4+2c+1}{c^6+c^3+4}$ and $c = x^p$. Our method consists then in finding which generalized exponents reduced mod p could match such a root s. Here, the result is that s can only correspond to the choice $\frac{4}{x+2} - 1$ at 3, $\frac{\beta^2}{(x-\beta)} + 3\beta$ at β and any choice at infinity. This means that in characteristic zero, a possible exponential solution must involve the generalized exponent $\frac{1}{36}\frac{\alpha^2}{(x-\alpha)} - \frac{1}{18}\alpha$ at α. This reduces the number of possible combinations from 81 to 3. Moreover, we have eliminated all "hard" combinations, *i.e.,* those for which the sum S in Lemma 1 would not be in $\mathbb{Q}(x)$.

If two generalized exponents differ by a rational number, then our mod p computation can not detect which one to take, which is an important limitation. This is why three combinations remained after the mod p computation even though there was only one root of $\chi_p(L)$. Of the three remaining combinations only one satisfies Fuchs' relation, and this combination yields an exponential solution.

2.2.3 Example 3

Let p be a prime number and consider the operator $L = x^4\partial^2 - px\partial - 2x - 1$. Then p is a good prime in the sense of Definition 46 below. However, despite the fact that $L[p]$

exists and can even be factored, neither of the two generalized exponents $3+1/p+p/x^2$ and $-1/p$ of L at $x=0$ can be reduced mod p. So both can be discarded by Lemma 4 and hence L has no exponential solutions.

2.3 Roots of $\chi_p(L)$ and τ-reduced terms

Definition 9. *If r is an object in characteristic zero we set $\tau(r) := \tau(r\,[p])$.*
Let $e_{x_i} \in \overline{\mathbb{Q}}[t_i^{-1}]$ be an unramified generalized exponent of $L \in \overline{\mathbb{Q}}(x)[\partial]$ at a point $x_i \in \overline{\mathbb{Q}} \cup \{\infty\}$. Let p be such that x_i and e_{x_i} can both be reduced mod p. The τ-reduced term corresponding to e_{x_i} *is then defined as*

$$\tau_{red}(e_{x_i}) := \begin{cases} \tau(\frac{e_{x_i}}{t_i}) & \text{if} \quad x_i \in \overline{\mathbb{Q}}, \\ -\tau(t_\infty e^*_\infty) & \text{if} \quad x_i = \infty \end{cases}$$

where e^*_∞ denotes e_∞ without its constant term.

Remark 4. *If $e \in \mathbb{Q}$ then $e[p] \in \mathbb{F}_p$ and Lemma 2 implies $\tau_{red}(e) = 0$.*

Definition 10. *A prime p is a* good prime *for L if the following conditions hold :*
$(C1)$- The coefficients of L can be reduced mod p *and the order of L does not decrease after reduction* mod p, i.e., *$a_n[p] \neq 0$.*
$(C2)$- All singularities can be reduced mod p.
$(C3)$- Distinct singularities stay distinct after reduction mod p.

If L is written in the form (2.3), then Condition $(C2)$ simply means "$\deg_x(a_n[p]) = \deg_x(a_n)$". And $(C3)$ means that the largest square-free factor of a_n remains square-free mod p. Checking if a prime p is good is fairly easy in practice. All three conditions are useful in our algorithm, for example, without $(C3)$ we would have to replace Algorithm CombMatchRoot in Section 2.5.2 by a more complicated algorithm. The lemma below uses $(C3)$ as well. Nevertheless, if p is not a good prime, then it is often still possible to obtain some useful information from a computation mod p, but we will not detail this.

Lemma 4. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$ and p be a good prime for L. If $\partial - r$ with $r \in \overline{\mathbb{Q}}(x)$ is a monic first order right hand factor of L and if $e_{x_i} \in \overline{\mathbb{Q}}[t_i^{-1}]$ is the generalized exponent of $\partial - r$ at x_i, then e_{x_i} can be reduced* mod p.

Proof. From Lemma 1, there exists a generalized exponent $e_{x_i} \in \overline{\mathbb{Q}}[t_i^{-1}]$ of L at each singularity x_i such that

$$r = \sum_{x_i \in \mathcal{S}} \frac{e_{x_i}}{t_i} - t_\infty e^*_\infty + \frac{Q'}{Q}, \tag{2.6}$$

where $Q \in \overline{\mathbb{Q}}[x]$ and e^*_∞ is e_∞ without its constant term. Since p satisfies $(C1)$, r can be reduced mod p (see Proposition 1). Now Q is defined up to a multiplicative constant (which we can choose in such a way that $Q[p]$ is defined and is not zero), so Q'/Q can be reduced mod p and hence the remaining part $\sum_{x_i \in \mathcal{S}} e_{x_i}/t_i - t_\infty e^*_\infty$ can be reduced mod p as well. Conditions $(C2)$+$(C3)$ in Definition 10 imply that we can recover $e^*_\infty[p]$

and $e_{x_i}[p]$ for $x_i \in \mathcal{S}$ from $(\sum_{x_i \in \mathcal{S}} e_{x_i}/t_i - t_\infty e^*_\infty)[p]$ and therefore $e_{x_i}[p]$ is well defined. Then the same is true for $e_\infty[p]$ by Fuchs' relation.□

We now give two new results linking the τ-reduced terms and the eigenvalues of the p-curvature in $\overline{\mathbb{F}}_p(x^p)$, *i.e.*, the elements of $\mathcal{R}(\chi_p(L))$. First another definition :

Definition 11. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$. We say that $s \in \mathcal{R}(\chi_p(L))$* comes from characteristic zero *if there exists $r \in \overline{\mathbb{Q}}(x)$ such that $\partial - r$ is a right hand factor of L in characteristic zero and $s = \tau(r)$.*

Proposition 2. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$ and let p be a good prime for L. If $s \in \mathcal{R}(\chi_p(L))$ comes from characteristic zero, then $\forall\, x_i \in \overline{\mathcal{S}}$, there exists a generalized exponent e_{x_i} at x_i such that :*

$$s = \sum_{x_i \in \overline{\mathcal{S}}} \tau_{red}(e_{x_i}).$$

Proof. By definition, $s \in \mathcal{R}(\chi_p(L))$ comes from characteristic zero means that there exists $r \in \overline{\mathbb{Q}}(x)$ such $\partial - r$ is a right hand factor of L. From Lemma 1, there exists then a generalized exponent $e_{x_i} \in \overline{\mathbb{Q}}[t_i^{-1}]$ of L at each singularity x_i such that r can be written in the form (2.6) where $Q \in \overline{\mathbb{Q}}[x]$ and e^*_∞ is e_∞ without its constant term. Since p is a good prime for L and Q is defined up to a multiplicative constant (which we can choose in such a way that $Q[p]$ is defined and is not zero), the reduction mod p of each term in Equation (2.6) is well defined (see also Lemma 4 above). Now using that $s = \tau(r)$, the result follows directly from Lemma 2 and the definition of the $\tau_{red}(e_{x_i})$.□

Proposition 3. *For almost all primes p the condition that s comes from characteristic 0 may be omitted.*

Proof. As we will not need this result for our algorithm, we will only sketch a proof. First we note that generalized exponents can be defined in characteristic p. Definitions 3 and 4 are completely algebraic : the operations required are additions and multiplications in $\overline{\mathbb{Q}}(x)$, tests to zero in $\overline{\mathbb{Q}}$ and computing the roots of a polynomial in $\overline{\mathbb{Q}}[x]$. Obviously these operations also make sense in characteristic p so the definitions are easily translated to characteristic p provided that the ramification k is not divisible by p (this is not an issue because we only consider $k = 1$ in this paper). The generalized exponents are here defined mod p, which is coherent with the fact that we want t_i^p to have generalized exponent 0 at x_i (because t_i^p is in the field of constants) as well as generalized exponent p at x_i because it vanishes with multiplicity p at x_i. Note that the notions of singularity, indicial equation and exponents in characteristic p have already been introduced in [Ho81] and [CC85].

Lemma 1 can also be adapted to characteristic p (the lemma is just a statement about the relation between a rational function and its local expansions). Then one can also give the analogue of Proposition 2 in characteristic p, with $L \in \overline{\mathbb{F}}_p(x)[\partial]$, with the x_i in $\overline{\mathbb{F}}_p \cup \{\infty\}$, the unramified generalized exponents e_{x_i} in $\overline{\mathbb{F}}_p[t_i^{-1}]$, but now s can be any root of $\chi_p(L)$.

Let $L \in \overline{\mathbb{Q}}(x)[\partial]$, let p be a good prime, and let s be any root of $\chi_p(L)$, not necessarily coming from characteristic 0. Applying the mod p analogue of Proposition 2 to $L[p]$ we see that even if s does not come from characteristic 0, the statement in Proposition 2 still holds if we replace the generalized exponents e_{x_i} of L by the generalized exponents of $L[p]$. This completes the proof because for all but finitely many good primes p, reduction mod p will induce a bijection between the (unramified) generalized exponents of L and those of $L[p]$.□

One obtains the following corollary which is related to the converse of [Ho81, Theorem 2] and [CC85, Corollary 1.4]. Indeed, their statement (attributed to Katz) is : if $L[p]$ has "sufficiently many solutions in a weak sense", which is equivalent to "L has nilpotent p-curvature", for almost all p, then all the exponents of L are rational numbers.

Corollary 2. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$. If all generalized exponents of L are rational numbers, then for almost all primes, $\mathcal{R}(\chi_p(L)) \subseteq \{0\}$.*

Proof. If $e_{x_i} \in \mathbb{Q}$ then $\tau_{red}(e_{x_i}) = 0$ and the result follows from Proposition 3.□

2.4 Exponential solutions and roots of $\chi_p(L)$

2.4.1 Classes of exponential solutions

Definition 12. *The* class $\overline{e_{x_i}}$ *of a generalized exponent e_{x_i} is $\overline{e_{x_i}} := e_{x_i} \mod \mathbb{Q}$, the equivalence class of e_{x_i} modulo $\mathbb{Q}$.*

Two generalized exponents e_{x_i} and e'_{x_i} that have the same class $\overline{e_{x_i}}$ and that can both be reduced mod p satisfy $\tau(e_{x_i}) = \tau(e'_{x_i})$ by Remark 4. So we can define $\tau(\overline{e_{x_i}}) := \tau(e_{x_i})$, and $\tau_{red}(\overline{e_{x_i}}) := \tau_{red}(e_{x_i})$. The corresponding definition for exponential solutions is :

Definition 13. *Let y be an exponential solution of L, $\partial - r$ be the associated first order right hand factor of L and e_{x_i} the associated generalized exponents in the sense of Lemma 1. Then the* class $\mathrm{Cl}(y)$ *of y is defined as the image of $r = y'/y$ in $\overline{\mathbb{Q}}(x)/\sim$ where $a \sim b \Leftrightarrow a - b = \frac{g'}{g}$ for some radical function $g \neq 0$. We can identify*

$$\mathrm{Cl}(y) = \sum_{x_i \in \mathcal{S}} \frac{\overline{e_{x_i}}}{t_i} - t_\infty e^*_\infty.$$

Two exponential solutions have the same class if and only if their quotient is a radical function, if and only if at every point, the associated generalized exponents have the same class. Now, to $L \in \overline{\mathbb{Q}}(x)[\partial]$ we associate the *class-set*

$$\mathrm{Cl}(L) := \{\, \mathrm{Cl}(y) \in \overline{\mathbb{Q}}(x)/\sim \mid y \text{ exponential solution of } L \,\}.$$

2.4.2 Links between $\mathrm{Cl}(L)$ and $\mathcal{R}(\chi_p(L))$

Lemma 5. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$ and p be a good prime for L. There is a map*

$$\begin{aligned} \Psi : \mathrm{Cl}(L) &\rightarrow \mathcal{R}(\chi_p(L)), \\ \mathrm{Cl}(y) &\mapsto \tau(y'/y) = \sum_{x_i \in \overline{\mathcal{S}}} \tau_{red}(\overline{e_{x_i}}). \end{aligned}$$

Proof. Let $\mathrm{Cl}(y) \in \mathrm{Cl}(L)$. An exponential solution y corresponds to a first order right hand factor $\partial - r$ of L in characteristic zero where $r = y'/y$. Since p is a good prime for L, this factorization can be reduced mod p (see Proposition 1) and we have thus a first order right hand factor $\partial - r[p]$ of $L[p]$. Now from Lemma 3, $\tau(r) \in \mathcal{R}(\chi_p(L))$. The lemma is then clear using Proposition 2 since $\tau(r)$ is an element of $\mathcal{R}(\chi_p(L))$ that comes from characteristic zero.□

Given L, there are two natural questions about Ψ :
$(Q1)$- Is Ψ surjective for almost all good primes ?
$(Q2)$- Does there exist a good prime p such that Ψ is injective ?

Proposition 4. *The answer to questions $(Q1)$ and $(Q2)$ can be no.*

Proof. We give one example for each question. For $(Q1)$, if we take a differential operator that has a basis of algebraic (but no exponential) solutions, then $\mathrm{Cl}(L) = \emptyset$. The proved part of Grothendieck's conjecture (see [Ka82], [Ch01] or [Pu01] for statements and histories of this conjecture) says that for almost all primes the p-curvature is zero so that $\mathcal{R}(\chi_p(L)) = \{0\}$. For $(Q2)$, take an operator L with the following 8 exponential solutions :

$$x^{e_0}(x-1)^{e_1}(x-2)^{e_2}$$

for $e_0 \in \{0, \sqrt{2}\}, e_1 \in \{0, \sqrt{3}\}$ and $e_2 \in \{0, \sqrt{6}\}$. Modulo every prime p, at least one of the three square roots becomes an element of $\mathbb{F}_p$. Suppose for example that $\sqrt{6}[p] \in \mathbb{F}_p$. Consider then the exponential solutions y_1 and y_2 corresponding to choices respectively $(0,0,0)$ and $(0,0,\sqrt{6})$ for (e_0, e_1, e_2). Then $\mathrm{Cl}(y_1) \neq \mathrm{Cl}(y_2)$ but they have the same image under Ψ.□

The previous proposition says that we can not always choose a prime number for which each root of $\chi_p(L)$ corresponds to at most one class of exponential solutions. The example illustrates the cause of this, and one can show the following (note that we do not use this in our algorithm) :

Proposition 5. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$. If there exists a good prime p for L for which :*

$$(*)\ \forall\, x_i \in \overline{\mathcal{S}},\ e_{x_i,2} - e_{x_i,1} \notin \mathbb{Q} \Rightarrow e_{x_i,2}[p] - e_{x_i,1}[p] \notin \mathbb{F}_p$$

then, for that prime, Ψ is injective.

Proof. Suppose that $\mathrm{Cl}(y_1) = \sum_{x_i \in \mathcal{S}} \frac{\overline{e_{x_i,1}}}{t_i} - t_\infty e^*_{\infty,1}$ and $\mathrm{Cl}(y_2) = \sum_{x_i \in \mathcal{S}} \frac{\overline{e_{x_i,2}}}{t_i} - t_\infty e^*_{\infty,2}$ are two distinct elements of $\mathrm{Cl}(L)$ and suppose that $\Psi(\mathrm{Cl}(y_1)) = \Psi(\mathrm{Cl}(y_2))$.

$$\begin{array}{rcl} \Psi(\mathrm{Cl}(y_1)) = \Psi(\mathrm{Cl}(y_2)) & \Leftrightarrow & \sum_{x_i \in \overline{\mathcal{S}}} \tau_{red}(e_{x_i,1}) = \sum_{x_i \in \overline{\mathcal{S}}} \tau_{red}(e_{x_i,2}), \\ & \Leftrightarrow & \tau(\sum_{x_i \in \mathcal{S}} \frac{\overline{e_{x_i,1}}}{t_i} - t_\infty e^*_{\infty,1}) = \tau(\sum_{x_i \in \mathcal{S}} \frac{\overline{e_{x_i,2}}}{t_i} - t_\infty e^*_{\infty,2}), \\ & \Leftrightarrow & \sum_{x_i \in \mathcal{S}} \frac{\overline{e_{x_i,2} - e_{x_i,1}}}{t_i} - t_\infty(e^*_{\infty,2} - e^*_{\infty,1}) = \frac{g'}{g}, \text{ for } g \in \overline{\mathbb{F}}_p(x)^*. \end{array}$$

This leads to : $\forall_{x_i \in \mathcal{S}}$, $e_{x_i,2}[p] - e_{x_i,1}[p] \in \mathbb{F}_p$ and $e^*_{\infty,2}[p] - e^*_{\infty,1}[p] = 0$ which, by Fuchs' relation, implies that also $e_{\infty,2}[p] - e_{\infty,1}[p] \in \mathbb{F}_p$. On the other hand, the hypothesis that $\mathrm{Cl}(y_1) \neq \mathrm{Cl}(y_2)$ means that $\exists\, i \in \overline{\mathcal{S}}, e_{x_i,2} - e_{x_i,1} \notin \mathbb{Q}$ which contradicts condition $(*)$.□

2.5 Modular improvements of the combinatorial problem

Definition 14. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$. L is said to be* rad-regular *if there exists $r \in \overline{\mathbb{Q}}(x)$ such that $L_{\partial \to \partial + r}$ has all its unramified generalized exponents in $\mathbb{Q}$. Otherwise, L is said to be* rad-singular *; it means that there exists a singularity x_i with two unramified generalized exponents whose difference is not in $\mathbb{Q}$.*

As mentioned in the introduction, we want to address two problems in Beke's algorithm : the combinatorial problem and the field problem. In this section we focus only on the combinatorial problem and not on the field problem ; we suppose that a number field C is given and develop an algorithm that finds all exponential solutions defined over C. We show how a careful modular analysis can effectively reduce the combinatorial problem if L is rad-singular. Unfortunately, in the rad-regular case no reduction in the combinatorial problem is obtained (see also Section 2.7.2).

2.5.1 Some precisions on the vocabulary

Let C be a number field. In light of Remark 1 we use the terminology :

Definition 15. *An exponential function y* is defined over C *if $y'/y \in C(x)$.*

When C is not algebraically closed, we will not consider all roots of the leading coefficient a_n of L, we will only consider one root for each irreducible factor over C. So the number of singularities that our algorithm considers is the number of irreducible factors of a_n (this number depends on the field C) plus one (the point at infinity). The only extensions of C that we will work with in Section 2.5 are extensions given by a single irreducible factor of a_n in $C[x]$, so Algorithm ExpSolsInC in Section 2.5.4 does not use computationally expensive nested extensions.

Definition 16. *A* place *is either a monic irreducible polynomial P_i in $C[x]$ or ∞.*
At each place $P_i \neq \infty$, we define the point x_i *as the class of x in $C[x]/(P_i)$ noted* $\mathrm{RootOf}(P_i)$. *At the place ∞, the point is also ∞.*

If the degree of P_i is one, then we can view x_i as an element of C. Recall that we have the local parameter $t_i = x - x_i$ at finite points and $t_i = 1/x$ at ∞. We adapt the notion of singularity and generalized exponents to the purpose of this section :

Definition 17. *The* C-singularities *of* $L \in C(x)[\partial]$ *written as in (2.3) are the points at the places* P_i *for all monic irreducible factors* P_i *of* a_n *in* $C[x]$ *and the point* ∞*. We keep the notations* $\mathcal{S}$ *for the set of finite* C*-singularities and* $\overline{\mathcal{S}} = \mathcal{S} \cup \{\infty\}$*. Furthermore, a* C-generalized exponent of L at x_i *is an unramified generalized exponent of* L *at the* C*-singularity* x_i *that further belongs to* $C(x_i)[t_i^{-1}]$*. The* degree *of a* C*-singularity* x_i *is the degree of* P_i *if* $x_i \neq \infty$*, and it is 1 if* $x_i = \infty$*.*

A first order operator $\partial - r \in C(x)[\partial]$ has a C-generalized exponent at every C-singularity. Hence, if at a C-singularity of L there are no C-generalized exponents, then L can not have exponential solutions defined over C.

Definition 18. *A* C-combination *is a list composed of one* C*-generalized exponent* e_{x_i} *at each* C*-singularity* $x_i \in \overline{\mathcal{S}}$*.*

The exponential solutions that we are looking for (those defined over C) match first order right hand factors $\partial - r$ with $r \in C(x)$ as in Lemma 1 but here we write :

$$r = \sum_{x_i \in \mathcal{S}} \mathrm{Tr}_i(\frac{e_{x_i}}{t_i}) - t_\infty e^*_\infty + \frac{Q'}{Q} \tag{2.7}$$

where Tr_i is the trace over the field extension $C(x_i, x) \supset C(x)$. The term $\mathrm{Tr}_i(e_{x_i}/t_i) \in C(x)$ is the sum of all conjugates of e_{x_i}/t_i over $C(x)$. So the above r is as in Lemma 1 with the additional property $r \in C(x)$, which means that for any σ in the Galois group of $\overline{\mathbb{Q}}(x)$ over $C(x)$, if we choose e_{x_i} at x_i in Lemma 1 then we must choose $\sigma(e_{x_i})$ at $\sigma(x_i)$ (σ maps the unramified generalized exponents of $L \in C(x)[\partial]$ at x_i to those at $\sigma(x_i)$). We extend Definition 9 to finite C-singularities x_i as follows :

$$\tau_{red}(e_{x_i}) := \tau(\mathrm{Tr}_i(\frac{e_{x_i}}{t_i})).$$

Definition 19. *Let* P *be a problem defined over a field* C *of characteristic 0 for which the following holds : if* s *is a solution of* P *defined over an algebraic extension of* C*, then all conjugates of* s *over* C *are solutions as well. Then we say that* $s_1, \ldots, s_k$ *are* the solutions of P up to conjugation over C *when :*

- for each s_i *a finite extension* C_i *of* C *is given such that* s_i *is defined over* C_i *and,*
- for any solution s *of* P *defined over any algebraic extension* C' *of* C *there is precisely one* i *such that* C_i *can be embedded in* C' *over* C *and* s_i *corresponds to* s *under this embedding.*

We define the degree *(or* algebraic degree*) of* s_i *over* C *as* $[C_i : C]$*. The total number of solutions defined over* $\overline{C}$ *equals the sum of the degrees over* C *of the* s_i*.*

Each such problem P that we need to solve (such as : find the singularities, find the generalized exponents at a singularity x_i, find the exponential solutions) will be solved

up to conjugation, this in order to prevent computationally expensive splitting fields. In this terminology, "computing the C-singularities" and "computing the singularities up to conjugation over C" have identical meaning : factor a_n over C and construct a field $C_i = C(x_i) = C[x]/(P_i)$ for each factor P_i (the singularity at infinity has degree 1 hence its field is C). Solving a problem up to conjugation over C does not involve constructing fields that contain more than one C_i, which means that we are not able to perform arithmetic between objects defined over distinct C_i's (unless of course when the degrees are 1, *i.e.*, when these C_i's are just C). That is why the e_{x_i}/t_i in equation (2.7) must first be "traced down" to an object defined over C before we can add them.

When we compute generalized exponents at x_i (recall that we only need the unramified generalized exponents), we compute them up to conjugation over $C(x_i)$. The C-generalized exponents are those of (algebraic) degree 1 over $C(x_i)$. Generalized exponents of degree > 1 will not be used in Section 5, they will be used in Algorithm FindASol in Section 2.6.2.

2.5.2 Finding C-combinations matching the modular information

Let $L \in C(x)[\partial]$ where C is a number field. Suppose that L has m finite C-singularities $x_1, \ldots, x_m$ and consider the following set :

$$\text{Comb} := \{(\overline{e_{x_1}}, \ldots, \overline{e_{x_m}}, \overline{e_\infty}) \,|\, e_{x_i} \text{ is a } C\text{-generalized exponent of } L \text{ at } x_i\}.$$

The elements of Comb are called *C-combinations mod $\mathbb{Q}$*. Note that the set Comb depends on C in two ways. First, the x_i correspond to irreducible factors of a_n over C. Second, the e_{x_i} must be defined over $C(x_i)$. We have an injective map $\text{Cl}(L) \to \text{Comb}$. From Proposition 2, if p is a good prime for L, we have a map from the set of the elements of $\mathcal{R}(\chi_p(L))$ that come from characteristic zero (the elements that interest us) to the set of subsets of Comb, or just to Comb itself if Condition $(*)$ in Proposition 5 holds. Consider now the map

$$\begin{array}{rccc} \Phi : & \text{Comb} & \to & \overline{\mathbb{F}}_p(x^p), \\ & (\overline{e_{x_1}}, \ldots, \overline{e_{x_m}}, \overline{e_\infty}) & \mapsto & \sum_{x_i \in \overline{\mathcal{S}}} \tau_{red}(\overline{e_{x_i}}). \end{array}$$

Definition 20. *Let $s \in \mathcal{R}(\chi_p(L))$ and $h \in$ Comb. We say that h* matches *s if $\Phi(h) = s$. We say that h* satisfies Fuchs' relation *if $\sum_{x_i \in \mathcal{S}} \text{Tr}_i(\text{Const}(\overline{e_{x_i}})) + \text{Const}(\overline{e_\infty}) = 0 \mod \mathbb{Q}$.*

Given an element s of $\mathcal{R}(\chi_p(L))$ for a good prime p, we can effectively find the finite set of elements h of Comb that match s. A naive way to do that consists in taking the image under Φ of all possible C-combinations mod $\mathbb{Q}$ and keeping those with image s. A reason for including Condition (C3) in the definition of a good prime is so that we can give a more practical method (Algorithm CombMatchRoot below).

In the sequel, $\mathbb{F} := C[p]$ denotes the image of the number field C after reduction mod p. In practice C is given by some (possibly nested) RootOf's so that this reduction

makes sense (recall that choosing a reduction $[p]$ means choosing factors mod p of the polynomials defining the RootOf's).

Since $\mathbb{F}$ is finite, the map $U \mapsto U^p$ is an isomorphism from $\mathbb{F}[x]$ to $\mathbb{F}[x^p]$, and we denote the inverse isomorphism by $V \mapsto V^{1/p}$. If $r = \frac{a}{b} + c$ with $a, b, c \in \mathbb{F}[x]$ and $\deg(a) < \deg(b)$ then $\tau(r)$ can be written as $\frac{A}{B} + C$ with $\deg(A) < \deg(B)$, $B = b^p$ (here $\gcd(A, B)$ need not be 1) and $C = \tau(c)$. So if $V \in \mathbb{F}[x^p]$ divides the denominator of $\tau(r)$ then $V^{1/p} \in \mathbb{F}[x]$ divides the denominator of r.

Algorithm CombMatchRoot
Input : a number field C, a good prime p for L, the reduction $[p]$, $s \in \mathcal{R}(\chi_p(L)) \cap \mathbb{F}(x^p)$ and the data structure of C-singularities and C-generalized exponents of L.
Output : the elements h of Comb matching s and satisfying Fuchs' relation.
1 - For each $x_i \in \overline{\mathcal{S}}$ and each e_{x_i} at x_i compute $\tau_{red}(e_{x_i})$.
2 - Compute the partial fraction decomposition $\sum_{i=1}^{N_s} \frac{U_i}{V_i^{n_i}} + W$ of s with U_i, $W \in \mathbb{F}[x^p]$, V_i irreducible in $\mathbb{F}[x^p]$ and $\deg(U_i) < \deg(V_i^{n_i})$.
3 - Remove all $\overline{e_\infty}$ for which $\tau_{red}(\overline{e_\infty}) \neq W$; if none remain, then return $\emptyset$ and stop.
4 - For each $i = 1 \dots N_s$,
  4*a* - Find the C-singularity $x_j = \text{RootOf}(P_j)$ such that $V_i^{1/p} \in \mathbb{F}[x]$ divides $P_j[p]$,
  4*b* - Remove all $\overline{e_{x_j}}$ for which the partial fraction decomposition of $\tau_{red}(\overline{e_{x_j}})$ does not contain the term $\frac{U_i}{V_i^{n_i}}$; if none remain, then return $\emptyset$ and stop.
5 - For all remaining $\overline{e_{x_j}}$, compute all combinations $h = (\overline{e_{x_1}}, \dots, \overline{e_{x_m}}, \overline{e_\infty})$ that satisfy Fuchs' relation and return those as output.

In step 1, if the reduction mod p fails then e_{x_i} can be discarded by the same argument as in Lemma 4. In steps 3 and 4*b*, if nothing remains, then there are no first order right hand factors $\partial - r$ of L with $r \in C(x)$ and $s = \tau(r)$. Condition $(C3)$ in the definition of good primes is used in step 4*a* to make sure that the P_j for which $V_i^{1/p}$ divides $P_j[p]$ is unique (the existence of such a P_j follows from the fact that a pole of a root of $\chi_p(L)$ is a singularity of $L[p]$).

Remark 5. *It follows from Lemma 4 that if e_{x_i} can not be reduced mod p then it can not be* relevant *for the algorithm, it can not correspond to any first order right hand factor. It follows from Algorithm* CombMatchRoot *above that a generalized exponent that can be reduced mod p but that does not match any element of $\mathcal{R}(\chi_p(L))$ can not be* relevant *either. If, adding these notions of relevant to those of Remark 2 (*i.e.*, unramified, $\mathbb{Z}$-minimal), no relevant generalized exponents remain at a certain singularity x_i then there are no exponential solutions and we can stop the computation. If at a certain x_i only one relevant generalized exponent remains then we call x_i a* semi-apparent singularity *because it does not contribute to the combinatorial problem. In Algorithm FindASol in Section 6 it is important that we use only relevant generalized exponents to prevent making unnecessary field extensions.*

Algorithm CombMatchMod$p$:
Input : an operator $L \in C(x)[\partial]$, a number field C, the data structure E of C-singularities and C-generalized exponents of L.
Output : the set of C-combinations mod $\mathbb{Q}$ that match a root of $\chi_p(L)$ for a good prime p and that satisfy Fuchs' relation.
1 - If at some C-singularity there are no C-generalized exponents, return $\emptyset$ and stop.
2 - Choose a good prime p for L (choose a reduction $[p]$).
3 - Compute $\mathcal{R}(\chi_p(L))$ and (to be used in Section 2.7) Nroots := the total number of roots of $\chi_p(L)$ in $\overline{\mathbb{F}}_p(x^p)$ counted with multiplicity.
4 - Return the union of CombMatchRoot(C,p,$[p]$,s,E) for $s \in \mathcal{R}(\chi_p(L)) \cap \mathbb{F}(x^p)$.

One can often (use Condition $(*)$ in Proposition 5) find a prime such that each s matches at most one C-combination mod $\mathbb{Q}$ (*i.e.*, the set CombMatchRoot(C,p,$[p]$,s,E) has ≤ 1 elements for each s). But Proposition 4 shows that such prime does not always exist so that, in practice, we do not try to find one.

2.5.3 Some remarks on the choice of p

We can choose p as follows. Start with $p = 2$, and as long as a reduction mod p of a coefficient of L fails, or if one of the other conditions of a good prime in Definition 10 is not met, we take the next prime. So we compute the smallest good prime. There are, however, certain cases where another choice could be better. In hard cases (when we need to use Algorithm FindASol, or when the number of combinations is very high) we may want to try a few more primes and select the best one. In very easy examples such as in Section 2.1 it can be best to use no primes at all because the combinatorial problem was already practically trivial without modular methods. Another situation where there is no improvement in the combinatorial problem is when L is rad-regular (see Definition 14 and also Section 7.2) and the p-curvature computation does not exclude existence of exponential solutions. In most cases we only use the smallest good p, and in the unusual event that this p is large then we skip the modular improvements given in Section 5 because p-curvature computations become expensive for large p. Exceptions are very easy cases (use no modular methods) and very hard cases (check more than one prime and select the best one).

2.5.4 Finding all exponential solutions defined over a given field

Algorithm CombMatchMod$p$ computes a set of possible combinations for C-generalized exponents mod $\mathbb{Q}$ that could appear in an exponential solution defined over C. We will use this to find all exponential solutions defined over C.

Recall that with the notations of Equation (2.7) and Lemma 1, an exponential solution can be written as $\exp(\int S)\, Q$ where $S = \sum_{x_i \in \mathcal{S}} \mathrm{Tr}_i(\frac{e_{x_i}}{t_i}) - t_\infty e^*_\infty$ and Q is a polynomial. After CombMatchMod$p$, it remains to check which C-combinations mod $\mathbb{Q}$

lead to exponential solutions and to compute these exponential solutions. A natural idea is : at each C-singularity, given a $\overline{e_{x_i}}$ we take the finitely many generalized exponents e_{x_j} satisfying $e_{x_j} \mod \mathbb{Q} = \overline{e_{x_i}}$ and we have thus a finite number of C-combinations $(e_{x_1}, \ldots, e_{x_m}, e_\infty)$, hence a finite number of possibilities for S after checking Fuchs' relation. Then for each S, we construct the operator $L_{\partial \to \partial + S}$ having as solutions those of L divided by $\exp(\int S)$, and we search for polynomial solutions Q of that operator.

We define an *E-combination* as a C-combination that uses only generalized exponents that appear in our data structure E. As already mentioned in Remark 2, replacing e_{x_i} by $e_{x_i} - d$ and Q by $t_i^d Q$ for some $d \in \mathbb{Z}$ leaves the expression $\exp(\int S)\, Q$ invariant. So we only need one generalized exponent in each equivalence class mod $\mathbb{Z}$. To assure that Q is a polynomial we only put $\mathbb{Z}$-minimal generalized exponents in our data structure E, *i.e.*, unramified generalized exponents that are minimal in their equivalence class mod $\mathbb{Z}$. This is also true for the point at infinity ; although the constant term of e_∞ does not contribute to S in Lemma 1, it does contribute in Equation (2.4) to the degree bound for Q (the number N in step 4 is used as a degree bound for the Q_i in step 4*b*), and using only $\mathbb{Z}$-minimal generalized exponents at infinity ensures that this degree bound is not too low.

Algorithm ExpSolsInC :
Input : an operator $L \in C(x)[\partial]$, a number field C.
Output : Sol, a basis of exponential solutions of L defined over C.
1 - Compute the data structure E of C-singularities and $\mathbb{Z}$-minimal C-generalized exponents.
2 - $M :=$ CombMatchMod$p(L,C,E)$.
3 - Sol $:= \emptyset$.
4 - For each E-combination $(e_{x_1}, \ldots, e_{x_m}, e_\infty)$ whose class mod $\mathbb{Q}$ is in M, and that satisfies :
$N := -\mathrm{Const}(e_\infty) - \sum_{x_i \in \mathcal{S}} \mathrm{Tr}_i(\mathrm{Const}(e_{x_i})) \in \mathbb{N}$, do :
4*a* - Let $S := \sum_{x_i \in \mathcal{S}} \mathrm{Tr}_i(\frac{e_{x_i}}{t_i}) - t_\infty e^*_\infty$ and $\tilde{L} := L_{\partial \to \partial + S}$,
4*b* - Compute a basis $Q_1, \ldots, Q_w$ of polynomial solutions in $C[x]$ for $\tilde{L}$,
4*c* - If $w > 0$ then, add the $\exp(\int S)\, Q_j$ to Sol.
5 - Return Sol.

We can pre-compute the $\mathrm{Tr}_i(\mathrm{Const}(e_{x_i}))$ and $\mathrm{Tr}_i(\frac{e_{x_i}}{t_i})$, from then on all the work in the loop in step 4 (which may dominate the computation time, see also Section 7.1) takes place over C.

To use this procedure in the recursive algorithm FindASol in Section 6.2, we need to allow two extra inputs. First, an option called "just one" ; if this option is given then we only need to return just one (if it exists) exponential solution defined over C but not a totally arbitrary one. We return a solution $\exp(\int S)\, Q$ with S as in the algorithm, but with Q of minimal possible degree. Second, an optional set F may be given in the input. Each element of F is a couple (x_i, e_{x_i}) where x_i is a C-singularity of degree 1 (see Definition 17) and e_{x_i} is a C-generalized exponent at x_i. This couple encodes the command : "in the data structure E in step 1 of ExpSolsInC, throw away all C-generalized exponents at x_i except e_{x_i}". This reduces the number of E-combinations to

be checked. Note that usually not all C-singularities will appear in F, so in general there is still a combinatorial problem. Furthermore, if $F \neq \emptyset$, then it is of course possible that no exponential solutions over C are found even if such solutions exist.

2.6 A way to handle the field problem

Let $L \in C(x)[\partial]$ with $C \subset \overline{\mathbb{Q}}$. We want to find a basis of all exponential solutions, so we may have to search for exponential solutions defined over algebraic extensions of C. Before constructing such extensions, we first want to bound their degrees.

2.6.1 Two ways to obtain bounds

We start with a few definitions, in particular we need the notion of *type*. Many useful properties of the type of a differential operator can be found in [Or32] (where type is called Art-Begriff). We will only use the type for first order operators, or equivalently, for exponential solutions :

Definition 21. *Two exponential functions y_1 and y_2 are said to be* of the same type *if y_1/y_2 is a rational function. This defines an equivalence relation, and* type(y_1)*, the* type *of y_1, is defined to be the equivalence class of y_1. We say that y_1 and y_2 have the same* local type *at x_i if y_1/y_2 is meromorphic at x_i, which is equivalent to saying that the generalized exponents of $\partial - y_1'/y_1$ and $\partial - y_2'/y_2$ at x_i differ by an integer.*

We will identify an exponential solution y with its minimal operator $\partial - y'/y \in \overline{\mathbb{Q}}(x)[\partial]$, and so we shall not distinguish y from a constant multiple of y. This makes it easy to describe what we mean by a *conjugate* of y ; if σ is an element of the Galois group of $\overline{\mathbb{Q}}$ over some number field C, then by $\sigma(y)$ we simply mean a non-zero solution of $\partial - \sigma(y'/y)$.

Two exponential functions have the same type if and only if they have the same local type at every point x_i. The type of the exponential solution $y = \exp(\int S)\, Q$ with S and Q as in Lemma 1 can be uniquely represented as $\sum_{x_i \in \mathcal{S}} \frac{e_{x_i} \mod \mathbb{Z}}{t_i} - t_\infty e_\infty^*$ where e_∞^* is e_∞ without its constant term (recall that this constant term depends on the constant terms of the other e_{x_i} by Fuchs' relation). The Galois group of $\overline{\mathbb{Q}}$ over C acts on the set of all types.

Definition 22. *Let C be a number field, y be an exponential function. Let K be the* field of definition *of y over C, which is defined as the smallest number field K that contains C for which $y'/y \in K(x)$. Then the* algebraic degree *of y over C is $[K : C]$. We say that y is of* minimal algebraic degree m over C *if $[K : C] = m$ and there exist no exponential function $\tilde{y}$, of the same type as y, having smaller algebraic degree over C.*

Let C be a number field, let y be an exponential function, and let K be the field of definition of y over C. Let $L_1 = \partial - y'/y \in K(x)[\partial]$. Let $L_1, \ldots, L_r \in \overline{\mathbb{Q}}(x)[\partial]$ be the conjugates of L_1 over C, and let $L = \mathrm{LCLM}(L_1, \ldots, L_r) \in C(x)[\partial]$. It is easy to

calculate a basis of all exponential solutions of L that have the same type as y ; compute $\tilde{L} := L_{\partial\to\partial+y'/y} \in K(x)[\partial]$, compute a basis $b_1, \ldots, b_k \in K(x)$ of rational solutions of $\tilde{L}$, then take $b_1 y, \ldots, b_k y$. Let $R \in K(x)[\partial]$ be the monic operator with $b_1 y, \ldots, b_k y$ as solutions. Now R is the unique monic operator whose solution space is the set of all solutions of L with the same type as y, so R is uniquely determined by two things : $L \in C(x)$ and type(y). Hence, any element σ of the Galois group of $\overline{\mathbb{Q}}$ over C that leaves type(y) invariant leaves R invariant as well. Conversely, if σ leaves R invariant, it leaves type(y) invariant as well because R has precisely one type of exponential solutions. Thus, one can define a *field of definition* of type(y) over C, namely as the smallest field C' that contains C for which $R \in C'(x)[\partial]$. Note that since $R \in K(x)[\partial]$, one has $C' \subseteq K$, hence : the field of definition of y over C contains the field of definition of type(y) over C.

We will now prove constructively that y is of minimal algebraic degree over C if and only if the fields of definition of y and type(y) are the same. To do this, we will show that R has an exponential solution defined over C', in fact, we will show that the algorithm ExpSolsInC with input R, C' will find such solutions. Since R has only one type, there is exactly one local type at each singularity x_i, so generalized exponents at x_i are unique modulo integers. So if σ is in the Galois group of $\overline{\mathbb{Q}}$ over $C'(x_i)$, then it can only move a generalized exponent e_{x_i} by an integer, but this implies that σ leaves e_{x_i} invariant, and thus $e_{x_i} \in C'(x_i)[t_i^{-1}]$, *i.e.*, e_{x_i} is a C'-generalized exponent. Hence there is exactly one $\mathbb{Z}$-minimal C'-generalized exponent at every singularity x_i, which implies that ExpSolsIn$C(R,C')$ will try exactly one C'-combination. This combination has the same type as y, and since all solutions of R have this type, ExpSolsInC will find a basis $y_1, \ldots, y_k$ of solutions of R, defined over C', where k is the order of R. We conclude the following :

Lemma 6. *Let C be a number field and $L \in C(x)[\partial]$. If y is an exponential solution of L defined over some algebraic extension K of C, then its conjugates over C are also solutions of L.*
Given L and y, we can compute $y_1, \ldots, y_k$ of minimal algebraic degree over C that form a basis of all exponential solutions of L of the same type as y.
If y is of minimal algebraic degree over C, then the distinct conjugates of y over C are of distinct type and hence linearly independent.

Proof. The fact that for $L \in C(x)[\partial]$, the conjugates over C of solutions are again solutions is clear.
From the foregoing, given L and y, we can compute an operator $R \in K(x)[\partial]$ whose solution space is precisely the set of all exponential solutions of L of the same type as y. We can calculate the smallest field C' containing C for which $R \in C'(x)[\partial]$, and can find a basis $y_1, \ldots, y_k$ of solutions of R with ExpSolsIn$C(R,C')$. Then $y_1, \ldots, y_k$ are defined over C', which is the field of definition of their type, so they have minimal algebraic degree over C. And they form a basis of all exponential solutions of L of the same type as y.

If σ is in the Galois group of $\overline{\mathbb{Q}}$ over C, and if two objects have the same field of definition over C, then σ leaves the first object invariant if and only if σ acts as the

identity on the field of definition, if and only if σ leaves the second object invariant. Now if y is of minimal algebraic degree over C, then it has the same field of definition as type(y), so σ leaves type(y) invariant if and only if σ leaves y (which we identify with $\partial - y'/y$ because we do not want to distinguish y from scalar multiples of y) invariant. Thus, distinct conjugates of y must have distinct types.□

Therefore, if we have an exponential solution of minimal algebraic degree m, then this exponential solution gives in fact m linearly independent solutions. In the output of our algorithm we will only return exponential solutions up to conjugation over C. In order to make it easier to count how many linearly independent solutions such an output represents, we will only allow exponential solutions in the output that have minimal algebraic degree over C.

Definition 23. *If C is a number field, x_i a singularity of L and e_{x_i} a generalized exponent at x_i, then* the field $C(x_i; e_{x_i})$ *is the extension of C given by x_i and the coefficients of e_{x_i}.*

Proposition 6. *Let C be a number field, $L \in C(x)[\partial]$, x_i a singularity, t_i the local parameter, and let $e \in \overline{\mathbb{Q}}[t_i^{-1}]$. Let b_1 be the number of distinct generalized exponents e_{x_i} of L at x_i with $e_{x_i} - e \in \mathbb{Z}$. Suppose that y is an exponential solution of minimal algebraic degree m over $C(x_i; e)$ with generalized exponent e at x_i. Then $m \leq b_1$.*

Proof. By the previous Lemma, the conjugates of y over $C(x_i; e)$ form m linearly independent exponential solutions, all of which have generalized exponent in $e + \mathbb{Z}$ at x_i. These exponential solutions (viewed as formal solutions at x_i) can be written $\phi_j(t_i) \exp(\int e/t_i \, \mathrm{d}t_i)$ where $\phi_j(t_i) \in \overline{\mathbb{Q}}((t_i))$ for all $j \in \{1, \ldots, m\}$. After Gaussian elimination on the vectors of coefficients of the $\phi_j(t_i)$, we obtain m formal solutions at x_i with distinct generalized exponents in $e + \mathbb{Z}$. Then $m \leq b_1$ because there are only b_1 such generalized exponents at x_i.□

The modular approach yields a different bound :

Proposition 7. *Let C be a number field, let $L \in C(x)[\partial]$ and let p be a good prime for L. Let b_2 be the number of roots of $\chi_p(L)$ (counted with multiplicity) that match (in the sense of Algorithm* CombMatchRoot*) the choice $\overline{e_{x_i}}$ at x_i. Let y be an exponential solution of minimal algebraic degree m over $C(x_i; e_{x_i})$, and generalized exponent e_{x_i} at x_i. Then $m \leq b_2$.*

Proof. Let $R \in C(x_i; e_{x_i})(x)[\partial]$ be the right hand factor whose solutions are spanned by all exponential solutions y of L that have generalized exponent e_{x_i} at x_i. Since there are at least m independent such solutions, the order d of R is at least m. All solutions of R have the same local type at x_i (namely the local type e_{x_i} mod $\mathbb{Z}$, see Definition 21). Now R factors as a product of first order factors $R = R_1 R_2 \cdots R_d$ because R has a basis of exponential solutions. All R_i have the same local type at x_i, so for each R_i, the generalized exponent at x_i must be congruent to e_{x_i} mod $\mathbb{Z}$. Hence the root of $\chi_p(R_i)$ matches $\overline{e_{x_i}}$ at x_i. The proposition now follows from the fact that R and $R_1, \ldots, R_d$ can

be reduced mod p (see Proposition 1) and Lemma 3(a).□

We now have two bounds b_1 and b_2 for the degree of the extension needed over $C(x_i; e_{x_i})$. We do not know in advance which is smaller, so we compute both and take the minimum.

Definition 24. *To each couple* (x_i, e_{x_i}) *we associate the bound* $b_{(x_i,e_{x_i})} := \min(b_1, b_2)$.

2.6.2 An algorithm to find an exponential solution over an algebraic extension

Let $L \in C(x)[\partial]$ with $C \subset \overline{\mathbb{Q}}$. We suppose that L has no exponential solutions defined over C and we want to find (if it exists) an exponential solution defined over an algebraic extension of C. We will use Algorithm ExpSolsInC that finds the exponential solutions of L defined over a given field, as well as the bounds from the previous section. To find an exponential solution over an algebraic extension of C, we increase the field step by step by adding extensions coming from singularities or generalized exponents, and at each step we use ExpSolsInC to search for an exponential solution over the new field. The bounds tell us when we can stop making field extensions.

For efficiency reasons we first want to reduce the order of the problem as much as possible before introducing field extensions, which is precisely what Algorithm ExpSols in Section 7 does before it calls Algorithm FindASol below. And since the order can be reduced whenever a solution is found, we want FindASol to stop computing as soon as it finds a solution.

In the algorithm we denote $e_{i,j}$, $j = 1, \ldots, n_i$ as all, up to conjugation over $C(x_i)$, see Definition 19, *relevant* (see Remarks 2 and 5) generalized exponents at x_i. If $n_i = 0$ then we can stop the algorithm, the output is $\emptyset$. Let F be a data structure as in the comments after Algorithm ExpSolsInC. If x_i appears in F, which means that a generalized exponent at x_i has already been chosen, then $n_i := 1$. We denote $d_{i,j} := [C(x_i; e_{i,j}) : C(x_i)]$, which is the degree as in Definition 19 of $e_{i,j}$ over $C(x_i)$. Now we define $b_{(x_i,e_{i,j},F)}$ as in Definition 24 with as difference that in Proposition 7 we only count those roots of $\chi_p(L)$ that match all generalized exponents specified in F. Of course, before we enter step 5 we discard all $e_{i,j}$ for which $b_{(x_i,e_{i,j},F)} = 0$ because we want n_i to be as small as possible, and if any n_i vanishes then we stop.

Algorithm FindASol :
Input : an operator $L \in C(x)[\partial]$, a field C, a data structure F and a positive integer b.
Output : an exponential solution of L or $\emptyset$.
1 - If $b = 0$, then return $\emptyset$ and stop.
2 - Run ExpSolsIn$C(L,C,$options=$\{$"just one", $F\}$).
3 - If this gives a solution, then return it and stop.
4 - If $b = 1$, then return $\emptyset$ and stop.
5 - If we can choose x_i in step 6 with degree 1 over C, or if $b' \leq b/2$ for all b' in step 6a, then go to step 6, else go to step 8.
6 - Choose one C-singularity x_i, not appearing in F. Then, for $j = 1, \ldots, n_i$ do 6a and 6b :
6a - Let b' be the minimum of $b_{(x_i,e_{i,j},F)}$ and $\lfloor \frac{b}{d_{i,j}} \rfloor$.
6b - If FindASol$(L, C(x_i; e_{i,j}), F \cup \{(x_i, e_{i,j})\}, b') \neq \emptyset$, then return it and stop.
7 - Return $\emptyset$ and stop.
8 - Do step 8a for all x_i not appearing in F for which $d_i := [C(x_i) : C]$ is not 1 and there are at least two j's with $d_{i,j} = 1$.
8a - If FindASol$(L, C(x_i), F, \lfloor b\frac{\lfloor d_i/2 \rfloor}{d_i} \rfloor) \neq \emptyset$, then return it and stop.
9 - For all i, j with x_i not appearing in F and $1 < d_{i,j} \leq b$ do step 9a :
9a - If FindASol$(L, C(x_i; e_{i,j}), F \cup \{(x_i, e_{i,j})\}, \lfloor \frac{b}{d_{i,j}} \rfloor) \neq \emptyset$, then return it and stop.
10 - Return $\emptyset$.

For the modular computations in recursive calls of ExpSolsInC, we always use the same prime number so that the modular information that we need is only computed once. One can also re-use information on singularities and generalized exponents from previous computations, but this information needs to be updated to the new field which involves factoring polynomials. For the singularities this means : factoring a_n over the new field. The generalized exponents need to be updated as well, for example, if we extend C then the degree over $C(x_i)$ of a generalized exponent can decrease, which means that the polynomials defining the field extension $C(x_i; e_{i,j})$ over $C(x_i)$ have become reducible, so they need to be factored. This way one conjugacy class of generalized exponents can split up into several conjugacy classes of generalized exponents.

We are going to verify (we must prove this recursively, *i.e.*, by induction) that FindASol satisfies the following :

Algorithm specification : if an exponential solution y exists that satisfies the fixed choices in F and has degree $\leq b$ over C, then FindASol will return some exponential solution of L (but not necessarily of degree $\leq b$ over C).

Proof. We first show that step 6 is correct. Suppose such y exists. Then the conjugates of y over C are also solutions of L, and satisfy the same fixed choices made in F (because in F we only have singularities and generalized exponents defined over C) as well as the degree bound b. Now the $e_{i,j}$ constitute, up to conjugation, a complete list of relevant $\mathbb{Z}$-minimal generalized exponents at x_i, so there must exist j such that a conjugate of y has generalized exponent in $e_{i,j}+\mathbb{Z}$ at x_i. We may assume that conjugate is just y. So for this j, there is a solution y defined over a field C' with $[C' : C] \leq b$ having generalized exponent in $e_{i,j} + \mathbb{Z}$ at x_i. We now replace C by $C(x_i; e_{i,j})$ and F by $F \cup \{(x_i, e_{i,j})\}$ so

we may use the degree bounds from the previous section. We included a new bound in the algorithm as well, namely $\lfloor \frac{b}{d_{i,j}} \rfloor$, which holds because :

$e_{i,j}$ is a C'-generalized exponent since it corresponds (mod $\mathbb{Z}$) to the solution y defined over C'. So $C'(x_i) = C'(x_i; e_{i,j})$. Let $\tilde{b} = [C'(x_i) : C(x_i)]$. Now $C'(x_i) = C'(x_i; e_{i,j})$ contains $C(x_i; e_{i,j})$, which has degree $d_{i,j}$ over $C(x_i)$, and so $[C'(x_i) : C(x_i; e_{i,j})] = \tilde{b}/d_{i,j}$. Now y is defined over C', hence over $C'(x_i)$, which is an extension of $C(x_i; e_{i,j})$ of degree $\tilde{b}/d_{i,j}$ and this is $\leq \lfloor \frac{b}{d_{i,j}} \rfloor$ because $\tilde{b} \leq [C' : C] \leq b$.

Now steps 8 and 9. Again suppose that an exponential solution y exists that satisfies the fixed choices in F and has degree $\leq b$ over C. Let again C' be the field of definition of y over C. If every C'-combination was also a C-combination then ExpSolsIn$C(L, C)$ and ExpSolsIn$C(L, C')$ would have to find an equal number of solutions; so FindASol will return a solution in step 3. Thus we may assume that not every C'-combination is a C-combination, so there are more C'-combinations than C-combinations. This can only happen when at least one of the following is true :

(I). There are more C'-singularities than C-singularities, or :
(II). There exists a C-singularity x_i at which there are more C'-generalized exponents than C-generalized exponents.

Case (I) will be handled by step 8 and case (II) by step 9. Suppose that (I) holds. Then the minimal polynomial of some C-singularity x_i must be reducible over C'. This minimal polynomial has degree d_i. One of the factors over C' must have degree $d' \leq \lfloor d_i/2 \rfloor$. By abuse of notation we denote x_i as a root of that factor as well. So then $[C'(x_i) : C'] = d'$, $[C' : C] \leq b$, so $[C'(x_i) : C] \leq d'b$. Now $C(x_i)$ is a field between $C'(x_i)$ and C having degree d_i over C, so $[C'(x_i) : C(x_i)] = [C'(x_i) : C]/d_i \leq d'b/d_i \leq b\epsilon$ where $\epsilon := \lfloor d_i/2 \rfloor / d_i$. Note that $\epsilon = 1/2$ when d_i is even, and $\epsilon < 1/2$ when d_i is odd. Now y is defined over C', hence also over $C'(x_i)$, which is an extension of degree $\leq b\epsilon$ over our new field $C(x_i)$. Hence the degree bound used in step 8a is correct.

Note that if there exists an i for which there is no j with $d_{i,j} = 1$ then a solution y necessarily involves a generalized exponent at x_i that is not defined over $C(x_i)$, the case that is treated by step 9, and in this case the algorithm stays correct (but becomes more efficient) if we skip 8 and use only this i in step 9. And if there is only one j with $d_{i,j} = 1$, then either we must choose this j (at x_i and its conjugates over C), or the problem is handled by step 9 (in which case it is enough to use only this i in 9).

Increasing the number of singularities does not increase the number of combinations if there is only one choice to be made at these singularities, thus skipping i in step 8 when there is only one j with $d_{i,j} = 1$ is correct.

Now assume that (I) does not hold (then every C'-singularity is a C-singularity) and assume that (II) holds. Then at some C-singularity x_i we have a C'-generalized exponent e_{x_i} that is not a C-generalized exponent. Say that this generalized exponent is defined over $C'(x_i)$ but has degree $d_{i,j}$ over $C(x_i)$. The degree bound $\lfloor \frac{b}{d_{i,j}} \rfloor$ in step 9a can then be proved with the same arguments as before, which completes the proof that FindASol satisfies its specification.□

The degree bound drops at least a factor 2 each time we use recursion with an

extension of C. So if the initial degree bound b is less than 4 then the extensions we construct are not greater than those given by a single x_i (in step 8) or a single couple (x_i, e_{x_i}) in step 9. If $b < 8$ then we will stack at most 2 such extensions (each given by an x_i or by a couple (x_i, e_{x_i})) on top of each other, which is generally feasible in practice. So unless the degree bound is very high we do not expect the algorithm to choke on these algebraic extensions. And even if the degree bound is very high (order $\geq$ Nroots ≥ 8), the alternatives currently used by Maple's *expsols* need not be better.

We can further improve the algorithm with modular methods. For example, certain x_i in step 8 may be omitted on the basis of modular information, namely when none of the roots of $\chi_p(L)$ allows a combination that uses distinct generalized exponents at two singularities conjugated to x_i.

If a root of $\chi_p(L)$ can only match a solution defined over C (this can happen when it must use an $e_{i,j}$ for which $C(x_i; e_{i,j}) = C$ and the bound in Definition 24 is 1) then we can discard this root after step 2. Reducing the number of roots of $\chi_p(L)$ in this way reduces the bounds and the n_i's which may help prevent making some unnecessary field extensions.

Step 8*a* can be improved in the following way. If the recursive call for x_i returns $\emptyset$, then later recursive calls in step 8*a* should be prevented from factoring the minimal polynomial of x_i over the new fields, and prevented from making that same extension x_i again.

A similar improvement is possible in step 6*b*. If the recursive call for $e_{i,j}$ returns $\emptyset$, then we could discard $e_{i,j}$. This will prevent later recursive calls from trying $e_{i,j}$ again. But it also reduces the number of generalized exponents, which may improve the bound b (see the number N_i in comment [1] after Algorithm ExpSols in Section 7).

2.7 An algorithm to find all exponential solutions

In this section, we give a complete algorithm to find a basis, up to conjugation, for all exponential solutions of a differential operator $L \in C(x)[\partial]$ where $C \subset \overline{\mathbb{Q}}$. The general idea in this recursive algorithm is the following. Using our procedure ExpSolsInC, we first compute the "easy" first order factors, that is, those defined over C. Then by looking at the value of Nroots (from Algorithm CombMatchMod$p$) we check if there may exist more exponential solutions, which are then necessarily not definable over C. If such exponential solutions could exist then we first recursively remove all "easy" factors, both on the right as well as on the left using the *adjoint* L^* of L (see [In26, 5.3] or [PS03, 2.1]) until there are no more easy factors on either side. We do this because we want to make the order of the operator as small as possible before entering the "hard" case (*i.e.*, searching for exponential solutions defined over algebraic extensions using Algorithm FindASol). The specifications of the following ExpSols algorithm are the following : first, it should not return the "same" solution more than once, more precisely : the output gives *only one* exponential solution in each conjugacy class. This is because we want the output to be a *basis up to conjugation*. Second, the solutions given in the output must all be of *minimal algebraic degree* over C, because this guarantees (see Lemma 10) that their conjugates over C are linearly independent. To count the number

of elements in the basis of exponential solutions, we must count a solution y given in the output $d(y)$ times, where $d(y)$ is the degree over C of the field of definition of y. In the following, Card(Sol) denotes the number of elements of the set Sol. The number of solutions represented by Sol is the sum of $d(y)$ taken over $y \in$ Sol. Finally, we mark with a [.] the steps for which explanations or comments are given after the algorithm.

Algorithm ExpSols :
Input : a linear differential operator $L \in C(x)[\partial]$ ($C \subset \overline{\mathbb{Q}}$) and the field C.
Output : Sol, a basis of exponential solutions of L up to conjugation over C.
– Sol := ExpSolsIn$C(L,C)$.
– If Card(Sol) $\geq$ Nroots -1, then return Sol and stop ([1]).
– If Sol $\neq \emptyset$ then,
- Write $L = \tilde{L}$ LCLM$(\partial - y'/y \,|\, y \in$ Sol),
- ExpSols$(\tilde{L},C)$ ([2]).
- Keep only the types not defined over C of the solutions found,
- For each remaining type represented by t ([3]),
. Compute a basis $R_1, \ldots, R_s$ of rational solutions of $L_{\partial \to \partial + t}$,
. Add the $\exp(\int t)\, R_i$ to Sol.
- Return Sol and stop.
– Else,
- If the order is 2 then go to the next Else below,
- Make L monic,
- ExpSolsIn$C(L^*,C)$ where L^* is the adjoint of L ([4]).
- If it finds solutions $y_1, \ldots, y_r$ where $r > 0$ then,
. Write $L =$ LCLM$(\partial - y_1'/y_1, \ldots, \partial - y_r'/y_r)^*\, \tilde{L}$,
. Return ExpSols$(\tilde{L},C)$.
- Else ([5]),
. FindASol$(L,C,\emptyset,$Nroots$)$ ([6]). If $\emptyset$ then return $\emptyset$ and stop.
. Optimize the solution y found ([7]),
. Compute a basis $R_1, \ldots, R_s$ of rational solutions of $L_{\partial \to \partial + y'/y}$,
. Add the $y\, R_i$ to Sol,
. Write $L = \tilde{L}$ LCLM$(\partial - r_1, \ldots, \partial - r_s,$ "and conjugates over C"), ([8])
. Remove recursively from $\tilde{L}$ the solutions of the same type as y ([9]),
. ExpSols$(\tilde{L},C)$.
. Keep only the types not defined over C of the solutions found,
. For each remaining type represented by t ([3]),
Compute a basis $R_1, \ldots, R_s$ of rational solutions of $L_{\partial \to \partial + t}$,
Add the $\exp(\int t)\, R_i$ to Sol.
. Return Sol and stop.

We explain or comment the points marked with a [.] :
[1] - The number Nroots, which was calculated in CombMatchMod$p$, is used as upper bound for the number of linearly independent exponential solutions. We have already found Card(Sol) independent solutions and so there are at most $d :=$ Nroots $-$ CardSol linearly independent exponential solutions left. This d is an upper bound for the degree of the field extension of remaining exponential solutions of minimal algebraic degree. If $d \leq 1$ then no extensions are necessary so we can stop the computation.
At a singularity x_i, let N_i be the number of distinct unramified generalized exponents in $\overline{\mathbb{Q}}[t_i^{-1}]$: this is another upper bound for the total number of linearly independent exponential solutions. If $N_i <$ Nroots, then replace Nroots by N_i.

[2] - At this point the roots of the p-curvature have already been computed, and need not be computed again ; we can simply take the roots (with multiplicity) for $\chi_p(L)$ and reduce the multiplicities according to the solutions found.
[3] - We have applied recursion on a left factor of L so the exponential solutions found are in general not exponential solutions of L. We will only use the types of the exponential solutions of this left factor, not the exponential solutions themselves. We skip types defined over C because all exponential solutions over C have already been found. From [Or32] or [Ho97b, Lemma 7.1], the types of exponential solutions of L that are not yet found must be among the types in the left factor $\tilde{L}$ of L.
[4] - We will apply FindASol only when all easy factors have been removed both on the left and on the right. Removal of easy factors, left or right, does not cause solutions to be lost because we only do this after all easy solutions have already been computed and stored in the set Sol, in the first step of the algorithm. To find the easy left factors, we apply ExpSolsInC to the adjoint operator L^* to find the easy exponential solutions of L^* which correspond to easy left factors of L (see [PS03, 2.1]). Then we apply recursion on the remaining right hand factor.
[5] - If the order of the operator is two, and if the bounds in FindASol do not immediately rule out a solution defined over an extension, then the formulas given in [UW96] would be a good alternative to FindASol. If the order is three, then an alternative would be the eigenring method (see [PS03]) but we do not expect that to be better than FindASol, especially if one implements some shortcuts for order 3 in FindASol (such as : we only need to consider extensions in FindASol that have a degree 3 subfield. And : if at x_i there are two non-conjugated generalized exponents that do not differ by an integer, then $C(x_i)$ is the only extension that needs to be considered).
[6] - Here, we reach the point where there is nothing left to do than entering the "hard case", trying to find a solution over an extension of C with FindASol. An *a priori* first bound is Nroots, the number of roots in $\overline{\mathbb{F}}_p(x^p)$ counted with multiplicity of $\chi_p(L)$ for a good prime p. Note that a more detailed analysis could lead to a sharper bound. For example, if two roots of $\chi_p(L)$ can not correspond to conjugated exponential solutions, or if one can conclude using the bounds in Section 2.6.1 that some root of $\chi_p(L)$ can not correspond to an exponential solution of minimal algebraic degree Nroots, then we could give FindASol a better bound.
[7] - Let y be an exponential solution. We can write $y'/y = P_1/P_2$ where P_1 and P_2 are polynomials with gcd 1 and P_2 is monic. The field of definition of y is then the field generated over C by the coefficients of P_1 and P_2. By "optimizing the solution" we mean two things : 1). Using this solution y to find a solution of minimal algebraic degree over C. And 2). Making sure that the field that the algorithm gives for y (this field contains the field of definition of y) is actually equal to the field of definition of y. Both 1) and 2) are important. We want y to be of minimal algebraic degree so that we know that its conjugates are linearly independent. But the way we count the number of conjugates of y, *i.e.*, the way we determine the number $d(y)$, is not by looking at y, but by looking at the field given for y. The field provided for y by Algorithm FindASol contains, but need not be not equal to, the field of definition of y. So to optimize y, we take the field given for y, and then determine the subfield generated over C by the

coefficients of P_1 and P_2. Then we find defining equations (*i.e.*, new RootOf's) for this subfield, and use them to rewrite the coefficients of P_1 and P_2. This then takes care of 2). To do 1), we could use the approach in Lemma 10, however, this is not necessary because the special choice that Algorithm ExpSolsInC makes when the option "just one" is given causes 1) to be automatically satisfied. The polynomial Q is of minimal degree with this option given, which leads to uniqueness of y'/y which in turn causes y to already be of minimal algebraic degree over C.
[8] - Here $r_i = R_i'/R_i + y'/y$. One can compute this LCLM using the method of undetermined coefficients and solving linear equations over $C(x)$, this can be done without actually constructing the conjugates of y'/y.
[9] - What is meant here is to also remove solutions whose type is conjugated over C to the type of y. We will remove such solutions recursively because if we do not then it would be non-trivial to ensure the specification of the algorithm that the "same" solution is not returned more than once. So before we call ExpSols on the remaining left factor $\tilde{L}$ of L, we first remove from $\tilde{L}$ all exponential solutions of the same type as (conjugates of) y, and we keep repeating this until $\tilde{L}$ no longer has solutions of this type. A recursive procedure to achieve that is the following : compute the rational solutions R_i of $\tilde{L}_{\partial\to\partial+y'/y}$; each one gives a first order right hand factor $\partial - r_i$ of L where $r_i = R_i'/R_i + y'/y$. If there are none then return $\tilde{L}$ else write $\tilde{L} = \tilde{L}_1$ LCLM($\partial - r_1, \ldots, \partial - r_s$, "and conjugates over C") and apply recursion on $\tilde{L}_1$.

2.7.1 Some remarks on the algorithm

Remark on ExpSolsInC : Even though we use Algorithm CombMatchMod$p$, there could still be combinations in step 4 that do not lead to an exponential solution (if for example the number of combinations is greater than Nroots then we already know in advance that this will be the case). If the degree bound (the number N in step 4) for polynomial solutions is high, or if C is a complicated field, or if there are many combinations to be checked, then to make the algorithm efficient we need a quick way to discard non-solution-combinations before doing all the work in steps 4a and 4b, because each of these two steps can dominate the computation time.

In step 4, we can of course pre-compute the traces, so that all computations in step 4 are rational (*i.e.*, over C). Computing S in step 4a then only involves additions in $C(x)$. However, that does not imply that computing S is cheap, because normalizing S (writing S as P_1/P_2 where P_1, P_2 are polynomials with no common factors) can be an expensive operation if C is a complicated field. Computation of $\tilde{L} = L_{\partial\to\partial+S}$ can be even more expensive because we need to multiply, differentiate, and add in $C(x)$. And since step 4a is applied to each combination, one can easily spend more time in step 4a than in steps 1 and 2 combined. And if N is large then 4b can take even more time.

Often the N's are very small, and one can improve the running time of ExpSolsInC substantially by implementing some cases for step 4b separately, such as the case $N = 0$ (in that case we should use a fast zero test for $\tilde{L}(1)$ without computing $\tilde{L}$ itself) and the case N small non-zero : here, one would be tempted to take an ansatz $\sum_{i=0}^{N} c_i x^i$, and to compute the coefficients of $\tilde{L}$(ansatz) which gives linear equations for the unknowns c_i.

But for small N and large $\tilde{L}$ this results in much more equations than we need because there are only $N+1$ unknowns. In this case, to make the ansatz approach efficient we should not fully evaluate $\tilde{L}(\text{ansatz})$ because then we computed much more equations than we need. Instead we should evaluate just enough coefficients of $\tilde{L}(\text{ansatz})$ (and do this without fully computing $\tilde{L}$ itself) so that we have enough equations to determine the c_i, and once the $c_i \in C$ are found then we finish with a quick zero test for $\tilde{L}(\sum_{i=0}^{N} c_i x^i)$. For large N we propose to compute $\tilde{L}$ and to use [ABP95].

If we find $w > 0$ polynomial solutions in step $4b$ then the computation time in $4a$ and $4b$ was well spent even if those steps took a lot of time. But when $w = 0$, and if C is a complicated field or N is a large number, then we want to avoid $4a$ and $4b$ with high probability. This is done as follows : take a prime number p (in general this p is not the same prime as we used for the p-curvature because for the p-curvature we only use small primes) such that p is greater than N plus the order of L, and also $C[p] = \mathbb{F}_p$ (such primes have density at least $1/[C : \mathbb{Q}]$ so they are easy to find). Then compute $\tilde{L}[p] \in \mathbb{F}_p(x)[\partial]$ and check if it allows a polynomial solution of degree $\leq N$. If not, skip $4a$ and $4b$.

More general fields : It is not difficult to generalize our algorithm to arbitrary fields of characteristic 0. We may assume that C is the field generated by the coefficients of L, so then C is a finitely generated extension of $\mathbb{Q}$, say $C = \mathbb{Q}(t_1, \ldots, t_k)[\alpha_1, \ldots, \alpha_l]$ where $t_1, \ldots, t_k$ are algebraically independent and the α_i are algebraic over $\mathbb{Q}(t_1, \ldots, t_k)$. Then the reduction mod p works as follows : choose random values for t_i in $\mathbb{Q}$. This gives a map f_1 from a subring R_1 of $\mathbb{Q}(t_1, \ldots, t_k)$ to $\mathbb{Q}$. If not all coefficients of the minimal polynomial of α_1 are in R_1, then choose different random values for t_i. After that, the minimal polynomial of α_1 is mapped to a polynomial over $\mathbb{Q}$, we can choose an irreducible factor, a root α_1' of that factor, and map α_1 to α_1'. Do the same for $\alpha_2, \ldots, \alpha_l$. This way we have a map f_2 from a subring R_2 of C to the field $\mathbb{Q}(\alpha_1', \ldots, \alpha_l')$. We can then choose a reduction mod p in the same way as before, so we have a map g from a subring of $\mathbb{Q}(\alpha_1', \ldots, \alpha_l')$ to a finite field. Then let the map $[p]$ be the composition $g \circ f_2$ of these two maps, which will be defined on some subring R_3 of R_2. One can always find evaluations in $\mathbb{Q}$ for the t_i and a prime number p such that all elements of C that we want to reduce mod p are in R_3.

Logarithms : Generalized exponents correspond to formal solutions. Since we are only interested in exponential solutions, the formal solutions that contain a logarithm are not of interest to us. Because of this, we can disregard those generalized exponents that do not correspond to a formal solution without a logarithm. Note that in any given equivalence class mod $\mathbb{Z}$, this observation could eliminate some but never all unramified generalized exponents, so it does not help much for the combinatorial problem. However, this observation may reduce the number of distinct unramified generalized exponents in some equivalence classes mod $\mathbb{Z}$, which can reduce the bound in Proposition 6 as well as the number N_i in comment [1] after Algorithm ExpSols. This could speed up Algorithm FindASol. Note that if L has an exponential solution of minimal algebraic degree $n-1$ where n is the order of L, then the adjoint of L has an exponential solution defined

over C. Algorithm ExpSols removes this solution, hence, if ExpSols calls FindASol with degree bound (the number b in FindASol) $N_i = n - 1$ then we may use $n - 2$ instead.

Deciding which generalized exponents match formal solutions without logarithms can be done if we compute $d+1$ terms of the formal solutions, where d is the exponent-difference. We propose to do this computation only when ExpSols has to call FindASol. To speed up this computation, one could compute the formal solutions modulo a suitably large prime number, if a logarithm occurs then a logarithm must occur in characteristic zero as well.

A related implementation : Although we do not have an implementation for the algorithm ExpSols in this paper, there is an implementation in Maple 9 for the difference case, *i.e.*, a procedure that computes the hypergeometric solutions of a difference equation. This implementation follows some of the ideas in this paper and performs very well in practice (a paper concerning the difference case is planned). We expect our ExpSols to perform equally well.

2.7.2 Computing radical solutions

Computing radical solutions is almost the same problem as computing exponential solutions, with only a few differences :

1. One only uses generalized exponents that are in $\mathbb{Q}$, see Equation (2.5) and Remark 3.
2. τ_{red} maps rational numbers to zero, which implies that the only root of $\chi_p(L)$ that matters is 0, but also implies that there is no reduction in the combinatorial problem.

This means that for computing radical solutions, the only useful modular information is the multiplicity of the root 0 in $\chi_p(L)$, which we can use as a bound for the number of linearly independent radical solutions, hence as a bound for the degree of the field extension over which such solutions (of minimal algebraic degree) are defined.

To save computation time, we can compute this multiplicity without computing $\chi_p(L)$, in the following way : Take a good prime p. Then compute a basis $z_1, \ldots, z_s$ of rational solutions of $L[p]$. If $s = 0$ then return 0, otherwise write $L[p] = L_1 L_2$ where $z_1, \ldots, z_s$ is a basis of solutions of L_2, apply recursion to L_1, then add s to the result for L_1, and return the answer. We expect this to be faster than computing $\chi_p(L)$ because rational solutions of a differential operator in characteristic p can be computed very quickly, $\mathcal{O}(\max(l,p)^2 p)$ field operations where l is a bound on the degree of the coefficients of L, see [Cl03].

2.7.3 Almost all primes

There are many results known that hold for all but finitely many p. For example, it is easy to show that

Lemma 7. *For all but finitely many p, the number of linearly independent radical solutions of L is bounded by the dimension of the space of rational solutions of $L[p]$.*

Thus, for almost all p, we could have used the number s in the previous section as a bound. The reader may wonder why we did not do so. The problem here is that of the many results that are known for all but finitely many p, the only ones we can use for the algorithm are those for which a *small prime p can be exhibited* with little computation. This problem is also the reason that the only information we use from the p-curvature is its characteristic polynomial.

Chapitre 3

Computing Hypergeometric Solutions of Linear Recurrence Equations

Ce chapitre constitue un article soumis au journal *Applicable Algebra in Engineering, Communication and Computing* co-écrit avec Mark van Hoeij : le texte a été laissé dans sa langue d'origine.

Introduction

Let C be a field of characteristic zero and $\overline{C}$ its algebraic closure. In many examples, $C = \mathbb{Q}$, but C can also be a number field and may also have transcendental degree > 0 over $\mathbb{Q}$, as long as the standard algorithms (such as zero-test in C, factoring in $C[x]$, etc.) are available.

A *linear difference operator*

$$L = a_n\,\tau^n + a_{n-1}\,\tau^{n-1} + \cdots + a_0\,\tau^0, \tag{3.1}$$

with $a_i = a_i(x) \in \overline{C}(x)$ is an operator that acts on functions $u = u(x)$ in the following way :

$$L(u) = a_n(x)\,u(x+n) + a_{n-1}(x)\,u(x+n-1) + \cdots + a_0(x)\,u(x).$$

The set of all such operators is denoted by $\overline{C}(x)[\tau]$. This is a non-commutative ring ; multiplication is given by composition. We will always assume that the *leading coefficient* a_n and the *trailing coefficient* a_0 are both non-zero. In this case, n is called the *order* of L. The *field of definition of* L is the smallest field $C' \subseteq \overline{C}$ such that $L \in C'(x)[\tau]$.

The linear difference operator L in (3.1) corresponds to the *linear recurrence equation (or relation)*

$$L(u) = 0, \quad \text{i.e.,} \quad a_n(x)\,u(x+n) + a_{n-1}(x)\,u(x+n-1) + \cdots + a_0(x)\,u(x) = 0. \tag{3.2}$$

A *solution* of L is a function u for which $L(u) = 0$. A *hypergeometric solution* is a solution u for which $u(x+1)/u(x)$ is a rational function, *i.e.*, an element of $\overline{C}(x)$.

Computing a hypergeometric solution u is equivalent to computing a first order factor $\tau - u(x+1)/u(x)$ of L in $\overline{C}(x)[\tau]$.

An algorithm for computing hypergeometric solutions of linear recurrence relations with polynomial coefficients was first given by Petkovšek in [Pe92]. This algorithm is very useful and has interesting applications. Unfortunately, in practice it is limited to the case where a_0 and a_n do not have complicated roots because the algorithm computes the splitting field of $a_0\, a_n$ and this can easily run out of control. Even when this splitting field is not too complicated, the algorithm is still not optimal because it searches through more combinations than necessary (see [Ho99, Example 1.5] and also [We01, Section 6.3]). To address these problems, the concept of *finite singularities* was introduced in [Ho99]. This notion and the associated Fuchs' relations allow to reduce the number of combinations to check and to mimic Beke's algorithm to find exponential solutions of linear differential equations (see [Be1894] or [CH04]). For $n < 4$, [Ho99, Theorems 2 and 3] show how to avoid splitting fields. In this paper we extend this to order $\geqslant 4$. We also give a number of additional improvements. We detail the steps of the algorithm and we show how they can be done efficiently. The complete HypSols algorithm is described in Section 9 and uses a similar approach as Algorithm ExpSols in [CH04] which computes the exponential solutions of linear differential equations. We will show in Section 10 how modular computations can be used to improve several steps in the algorithm. Finally, we will show how to construct an analogue algorithm for q-difference equations when q is not a root of unity. Note that a q-analogue of Petkovšek's algorithm is given in [APP98].

To summarize, the main contributions of the paper are :

- More details on how to implement the algorithm in [Ho99] efficiently.
- How to avoid splitting fields for arbitrary order (in [Ho99] this was done for order $\leqslant 3$).
- Modular improvements in Section 10.
- Introduction of local types and Fuchs' relations for q-difference equations in Section 12, so that an algorithm along the same lines can be given for the q-difference case as well.

The paper is organized as follows. Section 1 recalls the vocabulary concerning hypergeometric terms and their local types. Section 2 shows how the symmetric product is used in the algorithm. Sections 3 and 4 deals with the finite singularities : we show how the possible local types at these singularities can be computed efficiently. In Section 5 the same problem concerning the point at infinity is addressed. Section 6 discusses issues about field extensions. In Section 7 we give Algorithm ESols to compute the "easy" solutions and in Section 8 we describe Algorithm HSol that computes a "hard" hypergeometric solution addressing the field problem. The complete Algorithm HypSols is given in Section 9 which also lists a number of additional improvements. Section 10 details how modular computations can improve the algorithm. In Section 11, we give some timings. Finally, in the last section, we introduce the notions needed to construct an analogue algorithm for q-difference equations when q is not a root of unity.

3.1 Hypergeometric terms

Let C be a field of characteristic zero and $\overline{C}$ be its algebraic closure. In [PS97, Section 6.2], the authors prove the existence and uniqueness (up to difference isomorphisms) of a universal extension $\mathcal{V}$ for difference equations with coefficients in $\overline{C}(x)$. Consequently, given a difference operator L, we can view the solutions of $L(y) = 0$ as elements of this universal extension $\mathcal{V}$. The dimension of the solution space (the kernel of $L : \mathcal{V} \to \mathcal{V}$) is n.

One can view $\mathcal{V}$ as a subring of the ring $\mathcal{S}$ which is defined as follows

$$\mathcal{S} = \{[f] \mid f \in \overline{C}^{\mathbb{N}}\},$$

where $[f]$ is the equivalence class of all $\tilde{f} \in \overline{C}^{\mathbb{N}}$ for which $f - \tilde{f}$ has finite support. If f is a function that is defined on all but finitely many elements of $\mathbb{N}$, then its image $[f] \in \mathcal{S}$ is well defined. This way $\overline{C}(x)$ can be viewed as a subring of $\mathcal{S}$.

Definition 25. *A non-zero expression $u(x) \in \mathcal{V}$ is called a* hypergeometric term *if it satisfies $u(x+1)/u(x) \in \overline{C}(x)$. The rational function $r(x) = u(x+1)/u(x) \in \overline{C}(x)$ is then called the* certificate *of $u(x)$. It is the analogue of the* logarithmic derivative *from the differential case.*

Let $u(x)$ be a hypergeometric term, and view $u(x)$ as an element of $\mathcal{S}$. If $C = \mathbb{C}$ then we can use Gamma functions to represent $u(x)$ by a function, defined for sufficiently large integers, as follows :

$$c^x \, R(x) \prod_{i=1}^{k} \Gamma(x - \alpha_i)^{e_i},$$

where $c \in \mathbb{C}^*$, $R(x) \in \mathbb{C}(x)$ and $\forall\, i \in \{1, \ldots, k\}$, $\alpha_i \in \mathbb{C}$ and $e_i \in \mathbb{Z}$.

In this paper, we do not need to distinguish a hypergeometric term $u(x)$ from a constant times $u(x)$. Thus we can represent $u(x)$ by its certificate, which explains the following definition.

Definition 26. *Let C be a field of characteristic zero. A hypergeometric term $u(x)$ is said to be* defined over C *if $u(x+1)/u(x) \in C(x)$.*

Definition 27. *Let $u_1(x)$, $u_2(x)$ be hypergeometric terms, and $r_1(x) = u_1(x+1)/u_1(x)$, $r_2(x) = u_2(x+1)/u_2(x)$ be their certificates. Then $u_1(x)$ and $u_2(x)$ are said to be of* the same type *(or* similar*) if $u_1(x)/u_2(x) \in \overline{C}(x)$, or equivalently, if $r_1(x)/r_2(x)$ is the certificate of a rational function.*

Remark 6. *The following property (see [PWZ96, Proposition 5.6.2]) shows that this notion is very natural :*
u_1, u_2 are of the same type $\Leftrightarrow$ $u_1 + u_2$ is either a hypergeometric term or zero.

In fact, if $u_1, \ldots, u_k$ are hypergeometric terms of distinct types then they are $\overline{C}(x)$-linearly independent. Let $u(x)$ be a hypergeometric term defined over C and let $r(x) \in$

$C(x)$ be its certificate. We can factor $r(x)$ over C as follows. Note that in general C will not be algebraically closed, so the factors need not be linear.

$$r(x) = c\,\phi_1(x)^{e_1}\cdots\phi_m(x)^{e_m}, \tag{3.3}$$

with $c \in C^*$, $e_i \in \mathbb{Z}$, and $\phi_i(x) = x^{\deg(\phi_i)} + \phi_{i,1}x^{\deg(\phi_i)-1} + \cdots + \phi_{i,\deg(\phi_i)}x^0$, where $\phi_{i,j} \in C$, and $\phi_i \in C[x]$ is monic and irreducible over C. Because $r(x)$ is a rational function, the multiplicities e_i can be negative.

Denote $t := 1/x$. Now $r(x) = r(1/t)$ can be written as

$$c\,t^v\,(1 + d\,t + \mathcal{O}(t^2)), \tag{3.4}$$

where $c \in C^*$, $v \in \mathbb{Z}$ and $d \in C$. Equations (3.3) and (3.4) are related in the following way : the number c is the same, and furthermore :

$$v = -\sum_{i=1}^{m} e_i\,\deg(\phi_i) \qquad \text{and} \qquad d = \sum_{i=1}^{m} e_i\,\phi_{i,1}.$$

If we factor $r(x)$ over $\overline{C}$, so $r(x) = c\prod_i (x-\alpha_i)^{e_i}$, then we can write $v = -\sum_i e_i$ and $d = -\sum_i e_i\,\alpha_i$.

Definition 28. *With the notations above, the* local type *of the hypergeometric term $u(x)$ at $x = \infty$ is defined as*

$$g_\infty(u(x)) = (c, v, d + \mathbb{Z}) \in C^* \times \mathbb{Z} \times C/\mathbb{Z}.$$

If $g_\infty(u_1) = (c_1, v_1, d_1 + \mathbb{Z})$ and $g_\infty(u_2) = (c_2, v_2, d_2 + \mathbb{Z})$, then $g_\infty(u_1\,u_2) = (c_1\,c_2, v_1 + v_2, d_1 + d_2 + \mathbb{Z})$.

Example 1. *Let*

$$u(x) = 7^x \frac{\Gamma(x+\frac{1}{3})\Gamma(x+\frac{6}{5})^3}{\Gamma(x-\frac{2}{3})\Gamma(x-\frac{4}{5})} \tag{3.5}$$

be a hypergeometric term defined over $\mathbb{Q}$. Then

$$r(x) = \frac{u(x+1)}{u(x)} = 7\left(x+\frac{1}{3}\right)^1\left(x-\frac{2}{3}\right)^{-1}\left(x+\frac{6}{5}\right)^3\left(x-\frac{4}{5}\right)^{-1},$$

so that

$$c = 7, \quad v = -(1\cdot 1 - 1\cdot 1 + 3\cdot 1 - 1\cdot 1) = -2,$$

and

$$d = 1\cdot\frac{1}{3} - 1\cdot\left(-\frac{2}{3}\right) + 3\cdot\frac{6}{5} - 1\cdot\left(-\frac{4}{5}\right) + \mathbb{Z} = \frac{27}{5} + \mathbb{Z}.$$

Consequently,

$$g_\infty(u(x)) = \left(7, -2, \frac{27}{5} + \mathbb{Z}\right).$$

This could have been deduced from the expansion of $r(x)$ at infinity :

$$r(x) = r\left(\frac{1}{t}\right) = 7\,t^{-2}\,\left(1 + \frac{27}{5}\,t + \mathcal{O}(t^2)\right).$$

Definition 29. *Let $u(x)$ be a hypergeometric term defined over a field C of characteristic zero and let $r(x) = u(x+1)/u(x) \in C(x)$ be as in Equation (3.3). Let $q \in \overline{C}$ and let $p = q + \mathbb{Z}$ be the image of q in $\overline{C}/\mathbb{Z}$. Then the* local type *or* valuation growth *of $u(x)$ at the "point" p is defined as :*

$$g_p(u(x)) = \sum_i e_i \text{ where the sum is taken over all } i \text{ for which } \phi_i \text{ has a root in } p.$$

Note that $\phi_i(\pi) = 0$ for some $\pi \in p$ is equivalent to saying that $\phi_i(q + l) = 0$ for some $l \in \mathbb{Z}$.

Example 2. *Let $u(x)$ be as in Example 1.*
For $p = 2/3 + \mathbb{Z}$, we have $g_p(u(x)) = 1 - 1 = 0$ and for $p = 4/5 + \mathbb{Z}$, $g_p(u(x)) = 3 - 1 = 2$.
If p is any other element of $\overline{\mathbb{Q}}/\mathbb{Z}$, then $g_p(u(x)) = 0$.
So for this example we have calculated the local types of $u(x)$ for all $p \in \overline{\mathbb{Q}}/\mathbb{Z} \cup \{\infty\}$. We can verify now that the so-called Fuchs' relations *(see [Ho99, Equation 3]) are verified :*

$$v + \sum_{p \in \overline{\mathbb{Q}}/\mathbb{Z}} g_p(u(x)) = 0 \qquad \text{and} \qquad d + \sum_{p \in \overline{\mathbb{Q}}/\mathbb{Z}} g_p(u(x))\, p \equiv 0 \bmod \mathbb{Z}. \tag{3.6}$$

The first one is $-2 + 0 + 2 = 0$, and the second is $\frac{27}{5} + \left(0 \cdot \frac{2}{3} + 2 \cdot \frac{4}{5}\right) \equiv 0 \bmod \mathbb{Z}$.

The following statement is part of [Ho99, Theorem 1].

Theorem 1. *Let $u_1(x)$ and $u_2(x)$ be hypergeometric terms. Then $u_1(x)$ and $u_2(x)$ are of the same type if and only if they have the same local type at every $p \in \overline{C}/\mathbb{Z} \cup \{\infty\}$.*

This theorem is central to the algorithm. The way the algorithm uses it is as follows : from the definition of *type*, if we know the type of $u(x)$, then we know $u(x)$ up to some unknown factor in $\overline{C}(x)$. The theorem says that we know the type of $u(x)$ if and only if we know the local type of $u(x)$ at all "points" $p \in \overline{C}/\mathbb{Z} \cup \{\infty\}$. Hence, in order to find a hypergeometric solution $u(x)$ of a linear recurrence equation, we first have to find the type of $u(x)$, and in order to do that we first have to find the possible local types of $u(x)$ at each $p \in \overline{C}/\mathbb{Z} \cup \{\infty\}$. In Section 4 we compute for $p \in \overline{C}/\mathbb{Z}$ a finite set $\overline{g}_p(L)$ of *possible local types of L at p*. This means that for every hypergeometric solution u of L, the local type $g_p(u)$ of u at p is an element of $\overline{g}_p(L)$. In Section 5 we do the same for $p = \infty$.

3.2 The symmetric product

Definition 30. *Let L_1, $L_2 \in \overline{C}(x)[\tau]$. The* symmetric product *$L_1 \circledS L_2$ of L_1 and L_2 is defined as the monic operator $L \in \overline{C}(x)[\tau]$ of smallest order such that $L(u_1\, u_2) = 0$ for all u_1, $u_2 \in \mathcal{V}$ with $L_1(u_1) = 0$ and $L_2(u_2) = 0$.*

In this paper, we only use this when the order of L_2 is 1, and we give a formula for this case.

Let $L = \sum_{i=0}^{n} a_i \tau^i \in \overline{C}(x)[\tau]$ be a linear difference operator with $a_i \in \overline{C}(x)$ and let $r \in \overline{C}(x)$. Then

$$L_{\circledS}(\tau - r) = \frac{1}{b_n} \sum_{i=0}^{n} b_i \tau^i \text{ where } b_n = a_n \text{ and } \forall\, i \leq n-1,\ b_i(x) = a_i(x) \prod_{j=i}^{n-1} r(x+j).$$

We made the result monic (dividing by the leading coefficient b_n) in order to make $L_{\circledS}(\tau - r)$ uniquely defined. However, in the implementation, uniqueness is not necessary here, and we multiply $L_{\circledS}(\tau - r)$ by a polynomial in order to avoid computing with fractions.

One way that the symmetric product is used in the algorithm is as follows : suppose we have chosen a *candidate*, that is, for each point p we have chosen a possible local type of L at p (*i.e.*, an element of $\overline{g}_p(L)$). If this candidate does not satisfy Fuchs' relations given in Equation (3.6) then the candidate does not correspond to the type of any hypergeometric term $u(x)$ (see [Ho99, Theorem 1]), and thus it can be discarded. If it does satisfy these relations, then we have a *candidate type*. Then we apply a few tests in order to quickly discard those types for which it is easy to decide that L has no hypergeometric solution of that type. If the candidate type passes these tests, then we want to compute a basis $u_1, \ldots, u_m$ of all hypergeometric solutions of L of that type (if such solutions exist). We construct a function $r(x) \in \overline{C}(x)$ that represents this candidate type. This means that a non-zero solution u of $\tau - r$ has this type. So $u_1/u, \ldots, u_m/u$ must be in $\overline{C}(x)$. Now $1/u$ is a solution of $\tau - 1/r$ and so from Definition 30 it follows that

$$\frac{u_1}{u}, \ldots, \frac{u_m}{u} \text{ is a basis of rational solutions of } L_{\circledS}(\tau - \frac{1}{r}).$$

The calculation that leads to $\overline{g}_p(L)$ for $p \in \overline{C}/\mathbb{Z}$ in Section 4 is the same as the computation of the bounds in [Ho98]. By re-using this information, we can calculate the bounds for the rational solutions of $L_{\circledS}(\tau - 1/r)$ (bound for the denominator and for the degree of the numerator) before we even calculate the operator $L_{\circledS}(\tau - 1/r)$. This is useful, for example if the bound for the degree of the numerator is negative (this is one of the tests used) then we can discard the candidate type without computing $L_{\circledS}(\tau - 1/r)$. We do not need to run an algorithm for computing bounds (see [Abr95] or [Ho98]) for $L_{\circledS}(\tau - 1/r)$ because by re-using numbers already computed during the $\overline{g}_p(L)$ computation in Section 4 we can determine exactly the same bound as what would have been obtained with [Ho98].

So we need to compute rational solutions of $L_{\circledS}(\tau - 1/r)$, but since we already know a bound for the denominator and the degree of the numerator, we can construct $r(x)$ in such a way that we are reduced to finding polynomial solutions (with degree-bound already known) of $L_{\circledS}(\tau - 1/r)$, see Algorithm Search in Section 7. As mentioned, if the degree-bound is negative then the candidate type is discarded. If the degree-bound is 0 then we take the sum of the coefficients of $L_{\circledS}(\tau - 1/r)$ and do a zero-test (first a fast probabilistic test, then if necessary a full test). If the degree-bound is positive but not high, then we proceed by computing and solving necessary linear equations for

the coefficients of the polynomial solutions. Since we do not want to solve systems of equations with many variables, we should use a different method if the degree-bound is high (see [ABP95, Ba97]).

3.3 What does it mean to be a finite singularity ?

Definition 31 (Definition 8 in [Ho99]). *Let $L = a_n\tau^n + \cdots + a_0\tau^0 \in \overline{C}(x)[\tau]$ with $a_n \neq 0$ and $a_0 \neq 0$. After multiplying L on the left by a suitable element of $\overline{C}(x)$, we may assume that the coefficients a_i are in $\overline{C}[x]$ and $\gcd(a_0, \ldots, a_n) = 1$. Then $q \in \overline{C}$ is called a* problem point *of L if q is a root of the polynomial $a_0(x)\, a_n(x-n)$.*

We want a notion of singularities that is invariant under the shift operation $p \mapsto p+1$. Now p is invariant under the shift if $p = \infty$ (Section 5 discusses the singularity $p = \infty$) but not invariant if $p \in \overline{C}$. Thus, we will deviate from other definitions in the literature (our *problem points* are called *singularities* in [No29]) and define finite singularities as follows :

Definition 32. *$p \in \overline{C}/\mathbb{Z}$ is called a* finite singularity *of L if L has a problem point in p.*

Example 3. *Let*

$$L = x\left(x - \frac{1}{3}\right)\left(x + \frac{1}{4}\right)\tau^2 - \tau + x(x-3).$$

The order is $n = 2$. The roots of the trailing coefficient give the problem points 0 and 3, and the roots of the leading coefficient give the problem points $0+2$, $1/3+2$, $-1/4+2$. If $u(x) \in \mathcal{V}$ is a hypergeometric solution of L and if $q \in \overline{\mathbb{Q}}$ then

$$q\left(q - \frac{1}{3}\right)\left(q + \frac{1}{4}\right)u(q+2) - u(q+1) + q(q-3)u(q) = 0,$$

whenever this is defined. Solving for $u(q+2)$ gives

$$u(q+2) = -\frac{-u(q+1) + q(q-3)u(q)}{q(q-\frac{1}{3})(q+\frac{1}{4})}, \quad \text{in general} \quad u(q+n) = -\sum_{i=0}^{n-1} \frac{a_i(q)}{a_n(q)} u(q+i), \tag{3.7}$$

and solving for $u(q)$ leads to

$$u(q) = -\frac{q(q-\frac{1}{3})(q+\frac{1}{4})u(q+2) - u(q+1)}{q(q-3)}, \quad \text{in general} \quad u(q) = -\sum_{i=1}^{n} \frac{a_i(q)}{a_0(q)} u(q+i). \tag{3.8}$$

Formula (3.7) can be used to compute $u(q+2)$ from previous u-values if $q+2$ is not a problem point whereas Formula (3.8) can be used to compute $u(q)$ from next u-values if

q is not a problem point. Consequently the problem points of L are $\{0, 1/3, -1/4\} + 2 = \{2, 7/3, 7/4\}$ and $\{0, 3\}$. This means that the finite singularities are

$$\left\{2+\mathbb{Z}, \frac{7}{3}+\mathbb{Z}, \frac{7}{4}+\mathbb{Z}, 0+\mathbb{Z}, 3+\mathbb{Z}\right\} = \left\{\mathbb{Z}, \frac{1}{3}+\mathbb{Z}, \frac{3}{4}+\mathbb{Z}\right\},$$

so there are three finite singularities.

Consider $p = \mathbb{Z}$. If for example the values of $u(-5), u(-4)$ are given, then we can use the relations (3.7,3.8) to calculate $u(q)$ for $q \in \mathbb{Z}$ with $q < 2$. But $u(2)$ can not be computed because taking $q = 0$ in equation (3.7) leads to a division by zero. If $u(8), u(9)$ were given, then $u(q)$ could be computed for all $q \in \mathbb{Z}$ for which $q > 3$. We can not reach $u(3)$ from $u(8), u(9)$ because taking $q = 3$ in equation (3.8) results again in a division by zero.

The point $p = 1/3 + \mathbb{Z}$ is a finite singularity. If $u(5 + 1/3), u(6 + 1/3)$ were given, then $u(q)$ can be determined with equations (3.7,3.8) for all $q \in p$. However, if $u(-5 + 1/3), u(-4+1/3)$ were given then not all $u(q)$, $q \in p$ can be determined. So p is singular, but one only notices its singular behavior when going from the left to the right (with Equation (3.7)), and not when going from the right to the left (with Equation (3.8)). The point $p = 3/4 + \mathbb{Z}$ is a finite singularity for the same reason.

Any other element of $\overline{\mathbb{Q}}/\mathbb{Z}$ is not a singularity. For example, $p = 1/2 + \mathbb{Z}$ is not a finite singularity of L, because whenever we know any two consecutive values of u on p then the two formulas (3.7,3.8) allow us to go freely to the left and right without ever bumping into a division by zero ; there are no problem points in $p = 1/2 + \mathbb{Z}$.

In general, if p is a finite singularity, if n consecutive values $u(q+1), u(q+2), \ldots, u(q+n)$ are given, and if $q+1, \ldots, q+n$ are *on the left of the problem points*, then $u(q+l)$ can be computed for all integers $l \leqslant n$, but not always for all positive integers l because that would require passing through problem points. This is the essence of the concept of *left solutions of L at p* introduced in [Ho99] : they are solutions defined on a left half line. Similarly, if $q+1, \ldots, q+n$ were *on the right of the problem points*, then $u(q+l)$ could be determined for positive integers l, which means we have *right solutions of L at p*, they are defined on a right half line.

The set of all left (resp. right) solutions at p is denoted by $V_{p,l}(L)$ (resp. $V_{p,r}(L)$). For a precise definition of these sets, see [Ho99, Section 2]. Both are n-dimensional vector spaces, where n is the order of L (we always assume that a_n and a_0 are non-zero). So at a finite singularity p we distinguish two kinds of solutions, left and right solutions, which are separated by the problem points. If p is non-singular then $V_{p,l}(L)$ and $V_{p,r}(L)$ are not separated and so they can be identified.

Deforming the singularity p is a method to bypass the problem points, in order to go from the left to the right solutions, as well as from the right to the left solutions. The deformed operator L_ϵ is obtained by substituting $x \mapsto x + \epsilon$ in L. This deformation moves the singularity p to $p - \epsilon$, and turns p into a non-singular point : dividing by zero is now replaced by dividing by ϵ. Here ϵ is a new constant ; deforming increases our field of constants from $\overline{C}$ to $\overline{C}(\epsilon)$. Because of this change of constants, elements of the two solution spaces (left and right) of L at p can not be identified in a unique way with

solutions of the deformed operator L_ϵ, but we will see in Section 9.1 (see also [Ho99, Section 4.2]) that it is still possible to define useful maps between these solution spaces.

The main information we gather by going from the left to the right and from the right to the left in the deformed operator L_ϵ is the multiplicity of the factor ϵ that will appear in the denominator (or in the numerator). In the next section, the set $\overline{g}_p(L)$ is constructed from this data, and so is a bound for the denominators of the rational solutions to be computed in the algorithm (the same bound as in [Ho98]).

3.4 How to compute $\overline{g}_p(L)$ for finite p.

Let C be a field of characteristic zero and let $L \in C(x)[\tau]$ be a linear difference operator. Let $p \in \overline{C}/\mathbb{Z}$. In [Ho99] a procedure was given to calculate a finite set denoted by $\overline{g}_p(L)$ that has the property :

$$u(x) \text{ hypergeometric solution of } L \implies g_p(u(x)) \in \overline{g}_p(L). \tag{3.9}$$

Note : the converse does not hold in general.
We shall show how this can be done in practice in an efficient way. If $p = q + \mathbb{Z} \in \overline{C}/\mathbb{Z}$ where $q \in \overline{C}$ then computing with the singularity p of L is equivalent to computing with the singularity $0 + \mathbb{Z}$ of the operator L_q where L_q is the operator obtained from L by substituting $x+q$ for x. So without loss of generality we may assume that the singularity p to be considered is $p = \mathbb{Z}$. Note though that (see the discussion in Section 3.6), given $L \in C(x)[\tau]$ for some field C of characteristic zero, if $q \notin C$ then we need to temporarily (as long as we are computing with this singularity) replace the field C by $C' = C(q)$ because $L_q \in C'(x)[\tau]$.

Now we will assume that $L \in C'[x][\tau]$ has order n and that $p = \mathbb{Z}$ is the singularity to be studied. Then the problem points corresponding to p are the integer roots of $a_0(x)\, a_n(x-n)$. We take the largest possible $p_l \in p$ such that for all integers $q \leqslant p_l$, the two consecutive sequences $u(p_l - n), \ldots, u(p_l - 1)$ and $u(q-n), \ldots, u(q-1)$ can be computed from each other via the two equations (3.7,3.8) without bumping into divisions by zero. Then we can identify every *left solution* of L at p with a vector $(u(p_l - n), \ldots, u(p_l - 1))$ with entries in $\overline{C}$, because given any n consecutive values of such a left solution, we can determine $(u(p_l - n), \ldots, u(p_l - 1))$ by repeated use of Equation (3.7), and furthermore, given $(u(p_l - n), \ldots, u(p_l - 1))$ we can determine $u(q)$ for all integers $q < p_l$ by repeated use of Equation (3.8). Similarly, we can take the smallest integer $p_r \in p$ such that we can go back and forth between $(u(p_r+1), \ldots, u(p_r+n))$ and $(u(q+1), \ldots, u(q+n))$ for all $q \geqslant p_r$. This way each *right solution* of L at p can be identified with a vector $(u(p_r+1), \ldots, u(p_r+n))$ with entries in $\overline{C}$.

Remark 7. *The condition back* and *forth is important in order for such smallest $p_r \in p$ to exist. For example if $L = \tau^2 - x^2$ and $p = \mathbb{Z}$ then starting with $(u(p_r+1), u(p_r+2))$ and say $p_r < 0$ we can go arbitrarily far to the right, but say we go to $(u(1), u(2))$, then going from there to $(u(p_r+1), u(p_r+2))$ without dividing by zero is not possible when $p_r < 0$. So for $L = \tau^2 - x^2$ we can not take $p_r < 0$. We see that $p_r = 0$. For $L = \tau^2 - x^2$*

and $p = \mathbb{Z}$ we have $p_l = 2$. Notice that $p_l - n$ is always smaller than $p_r + 1$, but that $\{p_l - n, \ldots, p_l - 1\}$ and $\{p_r + 1, \ldots, p_r + n\}$ are not necessarily disjoint.

Example 4. *Back to Example 3, we recall that, concerning the finite singularity $p = \mathbb{Z}$, we have seen that we can not obtain $u(2)$ from previous u-values nor can we obtain $u(0)$ or $u(3)$ from next u-values. One finds $p_l = 2$ and $p_r = 3$.*

Since L_ϵ, the operator obtained by substituting $x+\epsilon$ for x in L, has no singularity at $p = \mathbb{Z}$, its solution space at p, $V_p(L_\epsilon) := \{\tilde{u} : p \to \overline{C}(\epsilon) \mid L_\epsilon(\tilde{u}) = 0\}$ is an n-dimensional $\overline{C}(\epsilon)$-vector space. We now specify two bases $\tilde{u}_1, \ldots, \tilde{u}_n$ and $\tilde{v}_1, \ldots, \tilde{v}_n$ of $V_p(L_\epsilon)$ in the following way :

$$\tilde{u}_i(p_l - j) = \delta_{i,j} \quad \text{and} \quad \tilde{v}_i(p_r + j) = \delta_{i,j} \quad \text{for} \quad i, j \in \{1, \ldots, n\},$$

where $\delta_{i,j}$ is 1 if $i = j$ and 0 otherwise. If we substitute $\epsilon = 0$ in $\tilde{u}_1, \ldots, \tilde{u}_n$ then one may have divisions by 0, but these do not occur at $q < p_l$. This way we obtain a basis $\{u_1, \ldots, u_n\} = \{\tilde{u}_1, \ldots, \tilde{u}_n\}_{\epsilon=0}$ of left solutions of L at p. Likewise, $\{\tilde{v}_1, \ldots, \tilde{v}_n\}_{\epsilon=0}$ is defined at all $q > p_r$, and this results in a basis of right solutions of L at p.

Definition 33. *Let $a \in \overline{C}(\epsilon)$. The* valuation $v_\epsilon(a)$ of a at $\epsilon = 0$ *is the element of $\mathbb{Z} \cup \{\infty\}$ defined as follows : if $a \neq 0$ then $v_\epsilon(a)$ is the largest integer $m \in \mathbb{Z}$ such that $a/\epsilon^m \in \overline{C}[[\epsilon]]$ and $v_\epsilon(0) = \infty$.*

So

$$v_\epsilon(a) > 0 \Longleftrightarrow a_{\epsilon=0} = 0 \quad \text{and} \quad v_\epsilon(a) < 0 \Longleftrightarrow a \text{ has a pole at } \epsilon = 0.$$

Now, for $p = \mathbb{Z}$, we want to compute a set $\overline{g}_p(L)$ satisfying the specification of (3.9). First, for $q \in \{p_l - n, \ldots, p_r + n\}$, we consider the following integers :

$$B^l(q) = \min_{1 \leqslant i \leqslant n} v_\epsilon(\tilde{u}_i(q)) \quad \text{and} \quad B^r(q) = \min_{1 \leqslant i \leqslant n} v_\epsilon(\tilde{v}_i(q)). \tag{3.10}$$

These integers are very useful since they also lead to bounds for the denominators of rational solutions in the same way as in [Ho98].

Definition 34. *The* minimum valuation growth $g_{p,r}(L)$ *and the* maximum valuation growth $g_{p,l}(L)$ *of L at p are defined as*

$$g_{p,r}(L) = \min\{B^l(q) \mid q > p_r\}, \quad g_{p,l}(L) = -\min\{B^r(q) \mid q < p_l\}, \tag{3.11}$$

and the set of valuation growths $\overline{g}_p(L)$ *is then*

$$\overline{g}_p(L) := \{m \in \mathbb{Z} \mid g_{p,r}(L) \leqslant m \leqslant g_{p,l}(L)\}.$$

Note on difference modules : If $p \in \overline{C}/Z$ and if two operators L_1, L_2 define isomorphic difference modules (for a definition see [PS97]) then $\overline{g}_p(L_1) = \overline{g}_p(L_2)$, and thus $\overline{g}_p$ can be defined for difference modules. This $\overline{g}_p$ behaves in a predictable way with respect to tensor products, direct sums, etc. Moreover, if $\overline{g}_p(L_1) = \{0\}$ then there

exists an operator L_2 defining an isomorphic difference module, for which p is not a singularity. We do not detail this because difference modules are not needed for this paper.

If $u = u(x)$ is a hypergeometric solution of L, then its *local type* $g_p(u)$ at p, also called its *valuation growth*, must be an element of this set $\overline{g}_p(L)$, see [Ho99]. If $\overline{g}_p(L)$ has only one element, then that leaves only one choice for $g_p(u)$. In that case p is called a *semi-apparent* singularity. If 0 is the only element of $\overline{g}_p(L)$ then p is called an *apparent* singularity. The singularities that are most significant in the algorithm are the ones where there is more than one choice for $g_p(u)$, *i.e.*, the singularities p where $\overline{g}_p(L)$ has more than one element. Such singularities are called *essential* singularities.

If L has a basis of rational solutions, then all singularities are apparent. More generally, if L has n linearly independent hypergeometric solutions, then $g_{p,r}(L)$ is precisely the smallest $g_p(u)$ of all hypergeometric solutions u of L, and $g_{p,l}(L)$ is the largest $g_p(u)$. If L has fewer than n linearly independent hypergeometric solutions then $g_{p,r}(L)$ and $g_{p,l}(L)$ are just bounds for $g_p(u)$ and are often no longer sharp. If $L = L_1 L_2$ then $\overline{g}_p(L_2) \subseteq \overline{g}_p(L)$ always holds, but it is easy to find examples where $\overline{g}_p(L_2) \neq \overline{g}_p(L)$, even when L does not have more hypergeometric solutions than L_2. This already occurs when $L = (\tau - x)(\tau - x) = \tau^2 - (2x+1)\tau + x^2$. Fortunately, this does not imply that the algorithm will have to construct many unnecessary operators $L_{\circledS}(\tau - 1/r)$ to check for rational solutions because Fuchs' relations (Equation (3.6)) are usually quite selective.
Now, in order to calculate the *minimum valuation growth* $g_{p,r}(L)$ from the formulas (3.10) and (3.11), we have to start from the $\tilde{u}_i$ (which take values 0 and 1 at $p_l - n, \ldots, p_l - 1$) and evaluate them at $p_r + 1, \ldots, p_r + n$. This computation starts with values at $p_l - n, \ldots, p_l - 1$, and then goes to the right using Equation (3.7).
Similarly, to compute the *maximum valuation growth* $g_{p,l}(L)$, we start with values 0 and 1 for $\tilde{v}_i$ at $p_r + 1, \ldots, p_r + n$, and then we proceed to the left (with Equation (3.8)) in order to evaluate $\tilde{v}_i$ at $p_l - n, \ldots, p_l - 1$.

Truncated power series :
Suppose we want to calculate $B^l(q)$ for $q \in \{p_l - n, \ldots, p_r + n\}$. We need to compute the $\tilde{u}_i(q)$ for $i \in \{1, \ldots, n\}$ and $q \in \{p_l, \ldots, p_r + n\}$. For efficiency reasons, we want to prevent computing in our new field of constants $C'(\epsilon)$. Recall that C' is the field extension of C given by the singularity $p \in \overline{C}/\mathbb{Z}$ that we shifted to $p = \mathbb{Z}$. Note that computing the sets $\overline{g}_p(L)$ using direct computations in $C'(\epsilon)$ hurts the running time of the algorithm : this is remarked in [We01, Section 6.3].
As we are merely interested in the ϵ-valuations of the $\tilde{u}_i(q)$, these $\tilde{u}_i(q) \in C'(\epsilon)$ can be replaced by truncated power series in ϵ, *i.e.*, power series of the form $\sum_{i<a} c_i \epsilon^i + \mathcal{O}(\epsilon^a)$ where the sum is finite and $a \in \mathbb{Z}$. This a is called *the accuracy* of the truncated power series. If $a_0(x), \ldots, a_n(x)$ are the coefficients of L then $a_0(x+\epsilon), \ldots, a_n(x+\epsilon)$ are the coefficients of L_ϵ. Let N_0 (resp. N_n) be the number of roots (counted with multiplicity) of $a_0(x)$ (resp. $a_n(x)$) in p. When we compute the next term with Equation (3.7), the accuracy decreases by the ϵ-valuation of $a_n(q+\epsilon)$, which is the same as the order of $a_n(x)$ at $x = q$. Combined, going from the left to the right, the accuracy decreases by N_n. We can recover the ϵ-valuation of an element of $C'(\epsilon)$ from a truncated power

series when the accuracy is greater than the ϵ-valuation. We start computing with $\tilde{u}(p_l - n), \ldots, \tilde{u}(p_l - 1)$ with initial accuracy $a = N_0 + N_n + 1$. Going to the right with Equation (3.7), the accuracy has decreased by N_n when we reach $\tilde{u}_i(q)$ with $q > p_r$, so the remaining accuracy is $a - N_n$. Considering the fact that one can go back with Equation (3.8), it is not difficult to show that N_0 is an a priori upper bound for $g_{p,r}(L)$, hence the remaining accuracy $a - N_n = N_0 + 1$ suffices to determine $g_{p,r}(L)$. The same initial accuracy $N_0 + N_n + 1$ suffices to compute $g_{p,l}(L)$ as well.

Example 5. *Let* $L = (x-1)\,(x+1)\,\tau^2 - x\,(x^2+x-1)\,\tau + x^2\,(x-1)$ *and* $p = \mathbb{Z}$. *The problem points are* 1, 3 *and* 0, 1. *Here* 1, 3 *are the solutions of* $a_n(x-n) = (x-1-2)\,(x+1-2) = 0$, *and* 0, 1 *are the roots of* a_0. *Now* $p_l - n, \ldots, p_l - 1$ *(the starting points for the* $\tilde{u}_i$*) is* -1, 0 *because this is the furthest to the right for which the following is true : for any* q, $q+1 \in p$ *on the left of* -1, 0 *one can go back and forth between* q, $q+1$ *and* -1, 0 *without dividing by zero. Likewise,* $p_r + 1, \ldots, p_r + n$ *(the starting points for the* $\tilde{v}_i$*) will be* 2, 3.

Replacing x *by* $x + \epsilon$ *to determine* L_ϵ, *and solving* $\tilde{u}(x+2)$ *from* $L_\epsilon(\tilde{u}) = 0$ *like in Equation (3.7) we get*

$$\tilde{u}(x+2) = -\frac{-(x+\epsilon)\,((x+\epsilon)^2 + (x+\epsilon) - 1)\,\tilde{u}(x+1) + (x+\epsilon)^2(x+\epsilon-1)\tilde{u}(x)}{(x+\epsilon-1)(x+\epsilon+1)}.$$

$N_0 + N_n + 1 = 6$ *so we start with :* $\tilde{u}_1(-1) = 0 + \mathcal{O}(\epsilon^6)$, $\tilde{u}_1(0) = 1 + \mathcal{O}(\epsilon^6)$. *Likewise, take* $\tilde{u}_2(-1) = 1 + \mathcal{O}(\epsilon^6)$, $\tilde{u}_2(0) = 0 + \mathcal{O}(\epsilon^6)$. *Then substitute* $x = -1, 0, 1$ *in the above equation to find :*

$$\begin{aligned}
\tilde{u}_1(1) &= -\tfrac{1}{2\epsilon} - \tfrac{1}{4} + \tfrac{7\epsilon}{8} - \tfrac{\epsilon^2}{16} - \tfrac{\epsilon^3}{32} - \tfrac{\epsilon^4}{64} + \mathcal{O}\left(\epsilon^5\right), \quad \tilde{u}_2(1) = -\tfrac{1}{\epsilon} + 2 - \epsilon + \mathcal{O}\left(\epsilon^5\right), \\
\tilde{u}_1(2) &= -\tfrac{1}{2} + \tfrac{\epsilon}{4} + \tfrac{\epsilon^2}{8} + \tfrac{9}{16}\epsilon^3 - \tfrac{23}{32}\epsilon^4 + \mathcal{O}\left(\epsilon^5\right), \quad \tilde{u}_2(2) = -1 + 3\,\epsilon - 3\,\epsilon^2 + 2\,\epsilon^3 - 2\,\epsilon^4 + \mathcal{O}\left(\epsilon^5\right), \\
\tilde{u}_1(3) &= -\tfrac{1}{4} - \tfrac{1}{4}\epsilon + \tfrac{3}{16}\epsilon^2 + \tfrac{11}{16}\epsilon^3 + \mathcal{O}\left(\epsilon^4\right), \qquad \tilde{u}_2(3) = -\tfrac{1}{2} + \tfrac{7}{4}\epsilon - \tfrac{3}{8}\epsilon^2 - \tfrac{13}{16}\epsilon^3 + \mathcal{O}\left(\epsilon^4\right).
\end{aligned}$$

The accuracy, indicated by $\mathcal{O}(\epsilon^a)$, *drops when we divide by* ϵ.
Now $B^l(q) = \min\{v_\epsilon(\tilde{u}_1(q)), v_\epsilon(\tilde{u}_2(q))\}$ *and* $g_{p,r}(L) = \min\{B^l(2), B^l(3)\}$ *so we have* $B^l(1) = -1$, $B^l(2) = 0$, $B^l(3) = 0$, *and* $g_{p,r}(L) = 0$.
Now use $L_\epsilon(\tilde{v}) = 0$ *to write* $\tilde{v}(x)$ *as a linear combination of* $\tilde{v}(x+1)$ *and* $\tilde{v}(x+2)$. *Then start with* $\tilde{v}_1(2) = 1 + \mathcal{O}(\epsilon^6)$, $\tilde{v}_1(3) = 0 + \mathcal{O}(\epsilon^6)$ *and* $\tilde{v}_2(2) = 0 + \mathcal{O}(\epsilon^6)$, $\tilde{v}_2(3) = 1 + \mathcal{O}(\epsilon^6)$ *and compute for* $i = 1, 2$ *the values of* $\tilde{v}_i(x)$ *at* $x = 1, 0, -1$. *At* $x = 1$ *we divide by* $a_0(1+\epsilon)$ *and the accuracy drops by 1. Then at* $x = 0$ *we divide by* $a_0(0+\epsilon)$ *and the accuracy drops by 2. So for* $i = 1, 2$ *we find* $\tilde{v}_i(1)$ *with accuracy 5, and find* $\tilde{v}_i(0)$ *and* $\tilde{v}_i(-1)$ *with accuracy* 3 *which is more than enough to determine* $g_{p,l}(L)$. *We find* $B^r(1) = -1$, $B^r(0) = -1$, $B^r(-1) = -1$ *and* $g_{p,l}(L) = -\min\{B^r(0), B^r(-1)\} = 1$. *Finally, we get*

$$\overline{g}_p(L) = \{m \in \mathbb{Z} \,|\, g_{p,r}(L) \leqslant m \leqslant g_{p,l}(L)\} = \{0, 1\}.$$

Our algorithm for computing hypergeometric solutions will try two candidate types *(see Section 2) for this example. In one of these candidate types, we have the element* 0 *from* $\overline{g}_p(L)$ *and the number* c *from Section 1 is 1. For this candidate type we need to find rational solutions of* L. *Now* $\max\{B^l(q), B^r(q)\}$ *is a lower bound for the valuation of*

a rational solution at $x = q$. This bound is -1 at $x = 1$ and zero elsewhere, thus, the rational solution must be of the form $(x-1)^{-1}$ times a polynomial. The degree bound for this polynomial is obtained from the information at $p = \infty$ computed in the next section. This degree bound is zero, and thus what remains to be done for this candidate type is to check if $(x-1)^{-1}$ is a solution, which it is.

The next candidate type has the element 1 from $\overline{g}_p(L)$ and the number c is again 1. So we have to compute the rational solutions of $L_{\circledS}(\tau - 1/r)$ where $\tau - r$ should have the desired candidate type, which implies that $r = x\frac{\tau(f)}{f}$ for some rational function f. Now the $B^l(q)$ number for $L_{\circledS}(\tau - 1/r)$ is the $B^l(q)$ number for L plus the $B^l(q)$ number for $\tau - 1/r$. The same holds for $B^r(q)$. We can choose f in $r = x\frac{\tau(f)}{f}$ in such a way that $L_{\circledS}(\tau - 1/r)$ will have $\max\{B^l(q), B^r(q)\} = 0$ for all $q \in \overline{C}$. This way, the rational solutions of $L_{\circledS}(\tau - 1/r)$ must be polynomials. The r obtained this way is $r = x - 1$. The degree bound for the polynomial solutions is zero, thus, the algorithm checks if the solution $\Gamma(x-1)$ of $\tau - r$ is a solution of L, which it is.

Example 6. *Let us see briefly what we obtain for the operator given in Example 3. In Example 4, we found $p_l = 2$ and $p_r = 3$, so we start with values of $\tilde{u}_i$ at 0, 1 and go to the right. We find $B^l(2) = B^l(3) = B^l(4) = B^l(5) = -1$ so that $g_{p,r}(L) = -1$. Then we start with values of $\tilde{v}_i$ at 4, 5 and go the the left, and we find $B^r(3) = B^r(2) = B^r(1) = -1$, $B^r(0) = -2$. So $g_{p,l}(L) = -\min\{-1, -2\} = 2$ and finally, $\overline{g}_p(L) = \{-1, 0, 1, 2\}$.*

Modular computations :
We can further improve this by computing modulo prime numbers. In our Maple implementation, this is done as follows : take three "big" primes p_1, p_2 and p_3, consider the coefficients of the truncated power series as elements of $\overline{\mathbb{F}}_{p_i}$ (details concerning the reduction of elements in C' modulo p are included in [CH04, Sections 1.3 and 7.1]) and make the computations modulo each p_i. If these three modular computations lead to the same result, then return it and stop, otherwise take new primes.

Remark 8. *With these two tricks, reducing rational functions in ϵ to truncated power series, and reducing the elements of C' to a finite field, computing the sets $\overline{g}_p(L)$ for $p \in \overline{C}/\mathbb{Z}$ is quite fast and does generally not dominate the running time of our algorithm.*

3.5 How to compute $\overline{g}_\infty(L)$.

As with finite points, given $L \in C(x)[\tau]$, one can exhibit a finite set $\overline{g}_\infty(L)$ such that :

$$u(x) \text{ hypergeometric solution of } L \Longrightarrow g_\infty(u(x)) \in \overline{g}_\infty(L).$$

We detail here how we can compute this set with the classical techniques (see [Ad28, Bi30, No29]) used to compute the formal solutions at infinity of linear difference operators. Our approach is based on the following easy lemma which follows directly from the definition of the local type at infinity and the exposition in [Ad28] or [Bi30].

Lemma 8. *Let $L \in C[x][\tau]$. If L has a hypergeometric solution $u = u(x)$ with local type $(c, v, d + \mathbb{Z})$ at infinity, then L has a formal solution at infinity of the form $\Gamma(x)^{-v}\, c^x\, x^d\, F(1/x)$ where $F(1/x)$ is a formal power series in $1/x$.*

The possible couples $(c, v) \in \overline{C} \times \mathbb{Z}$ appearing in the formal solutions at infinity can be read from the so-called *Newton τ-polygon* (see [BD94] and note that similar polygons already appeared in [Ad28, Bi30]) and more precisely $-v$ is a (integer) slope of this polygon whereas c is a root of the *Newton τ-polynomial* (or characteristic equation) associated with the slope $-v$.

Definition 35. *Let $L = \sum_{i=0}^{n} a_i\, \tau^i \in C[x][\tau]$. For $(u, v) \in \mathbb{R}^2$, let $\mathcal{L}(u, v) := \{(x, y) \in \mathbb{R}^2 \,|\, x = u,\, y \geqslant v\}$. Let $\mathcal{M}(L)$ be the union of the $\mathcal{L}(i, -\deg(a_i, x))$ for $i \in \{0, \ldots, n\}$ with $a_i \neq 0$. The* Newton τ-polygon of L *is the convex hull of $\mathcal{M}(L)$.*

We give now an algorithm that computes only relevant (*i.e.*, integer) slopes of the Newton τ-polygon with their associated Newton τ-polynomial.

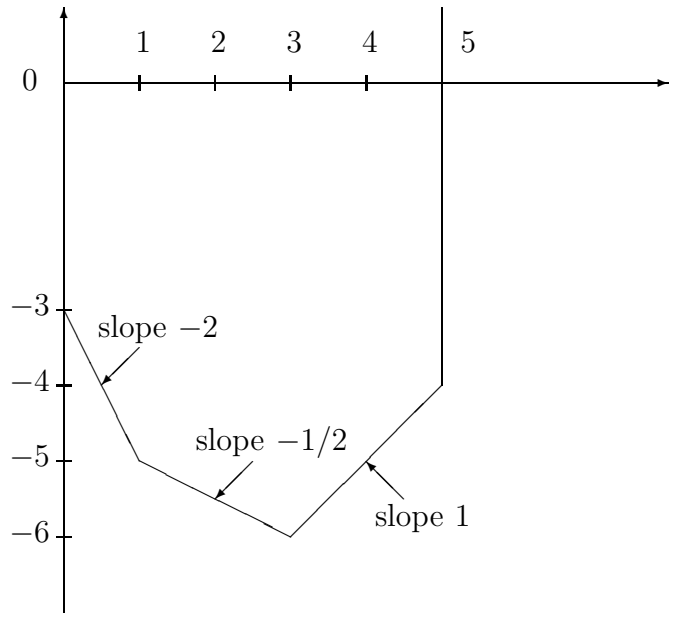
Algorithm τPolygon **:**

Input : $L := a_n\tau^n + \cdots + a_0\tau^0$ with $a_i \in C[x]$, a_0, $a_n \neq 0$.
Output : The integer slopes of the Newton τ-polygon and their corresponding Newton τ-polynomials.
1. If $n = 0$, then return $\emptyset$ and stop.
2. For $i \in \{0, \ldots, n\}$, let $v_i := -\deg(a_i, x)$ (if $a_i = 0$ then $v_i = \infty$).
3. $s := \max((v_n - v_i)/(n - i), i = 0, \ldots, n - 1)$.
4. $m := \min(i < n \,|\, (v_n - v_i)/(n - i) = s)$.
5. $R := \emptyset$.
6. If $s \in \mathbb{Z}$, then
  6a. $P := \sum_{i=0}^{n-m} \text{coefficient}(a_{n-i}, x^{s\,i-v_n})\, x^{n-m-i}$,
  6b. $R := \{[s, P]\}$.
7. Return $R \cup \tau\text{Polygon}(\ a_m\tau^m + \cdots + a_0\tau^0\ )$ and stop.

Example 7. *Let*

$$L := x^4\,\tau^5 + (5\,x^5 - 12\,x^3 - 3\,x)\,\tau^4 - (x^6 + x + 7)\,\tau^3 - (140\,x^3 + 1)\,\tau^2 + 10\,x^5\,\tau - 8\,x^3.$$

The Newton τ-polygon of L is :



Its slopes are 1, $-1/2$, -2.
Algorithm τPolygon *first finds the slope* $s = 1$ *and computes the associated Newton* τ*-polynomial* $P = x^2 + 5\,x - 1$. *Then, in the recursive call, considering only the terms of* L *of order* $\leqslant 3$ *(since* $m = 3$*), it finds* $s = -1/2$ *(with* $m = 1$*) which is not an integer. Thus, it continues with only the terms of* L *of order* $\leqslant 1$ *and finds the slope* $s = -2$ *(and* $m = 0$*) : it computes the associated Newton* τ*-polynomial* $P = 10\,x - 8$ *and terminates. Consequently, Algorithm* τPolygon *returns :* $\{[1, x^2 + 5\,x - 1], [-2, 10\,x - 8]\}$.

Note that the above algorithm also works more generally for $L \in C((t))[\tau]$ where $t = 1/x$ if we replace $-\deg(a_i, x)$ by $v_t(a_i)$ where v_t denotes the t-adic valuation.

If $\tau - r$ is a right hand factor of L then $\tau - r/(c\,t^v) = \tau - (1 + d\,t + \mathcal{O}(t^2))$ is a right hand factor of $L_{c,v}$ which is defined as (note : $1/t^v$ is just x^v)

$$L_{\circledS}(\tau - 1/(c\,t^v)),$$

multiplied on the left by a polynomial in order to avoid fractions. Now write $\delta = \tau - 1$ so $\tau = \delta + 1$ and $\tau - (1 + d\,t + \mathcal{O}(t^2)) = \delta - d\,t + \mathcal{O}(t^2)$. Let

$$L^\delta_{c,v} := \text{substitute}(\tau = \delta + 1, L_{c,v}) \in C(c)[x][\delta].$$

The transformation we have made to obtain $L_{c,v}$ amounts to dividing the formal solution in Lemma 8 by the term $\Gamma(x)^{-v}\,c^x$ so that the first order right hand factor $\delta - d\,t + \mathcal{O}(t^2)$ we are searching for corresponds to a formal solution of the form $t^{-d}\,F(t)$ where $F(t)$ is a formal power series in t with non-zero constant term.

Definition 36. *Let* $L \in C((t))[\delta]$ *and* e *be a variable. Let* $L_e := L(t^e)/t^e \in C[e]((t))$. *Then, we define* $v_1(L)$ *as the* t*-adic valuation of* L_e. *The* indicial equation of L *is the element of* $C[e]$ *defined as the coefficient of* $t^{v_1(L)}$ *in* L_e.

This definition is the one used in [AB98]. Note that, in [BD94, Section 2.1], a slightly different *indicial equation* is defined from another polygon we can associate to a difference operator : the *Newton δ-polygon*. Our indicial equation can be computed in the same way.

If $f \in C((t))^*$ has t-adic valuation $v_t(f)$, then the t-adic valuation of $L(f)$ is $v_t(f) + v_1(L)$ if $v_t(f)$ is not a root of the indicial equation and is $> v_t(f) + v_1(L)$ otherwise. Consequently, $v_1(L)$ can be viewed as the amount the t-adic valuation changes if we apply L to a "generic element" of $C((t))$. It is then easy to see that if L_1, $L_2 \in C((t))[\delta]$, then $v_1(L_1\, L_2) = v_1(L_1) + v_1(L_2)$ so that v_1 is a valuation on $C((t))[\delta]$.
Note that $v_1(t^i\, \delta^j) = i+j$, and that if $L = \sum_{i,j} a_{i,j}\, t^i\, \delta^j$ then $v_1(L) = \min(i+j \mid a_{i,j} \neq 0)$. So the only coefficients that contribute to the indicial equation are the $a_{i,j}$ for which $i + j = v_1(L)$. Set $e_j = (-1)^j\, e\, (e+1)\, (e+2) \cdots (e+j-1) \in C[e]$ then $\delta^j(t^e)/t^e = e_j\, t^j + \mathcal{O}(t^{j+1})$ so the indicial equation of $L = \sum_{i,j} a_{i,j}\, t^i\, \delta^j$ is the sum of the $a_{i,j}\, e_j$ taken over all i, j for which $i + j = v_1(L)$. From [Ad28, Bi30] we know that :

Lemma 9. *If $L \in \overline{C}((t))[\delta]$ has a solution of the form $t^e\, F(t)$ where $F(t)$ is a formal power series in t with non-zero constant term and $e \in \overline{C}$, then e is a root of the indicial equation of L.*

The roots of the indicial equation are called the *exponents* of L. The numbers d we are looking for are *negated exponents* $d = -e$.

Algorithm LocalTypeAt∞ **:**

Input : $L \in C(x)[\tau]$ and the field C.
Output : the set $\overline{g}_\infty(L)$, up to conjugation over C.
1. $\mathcal{A} := \emptyset$.
2. $\mathcal{P} := \tau$Polygon(L). If $\emptyset$, then return $\emptyset$.
3. For each $[-v, P] \in \mathcal{P}$, do :
  3a. For each (up to conjugation over C) root c of P, do :
    (i) $L_{c,v} := L_{\circledS}(\tau - x^v/c)$,
    (ii) Clear denominators in $L_{c,v}$,
    (iii) $L^\delta_{c,v} :=$ substitute$(\tau = \delta + 1,\, L_{c,v})$,
    (iv) Compute the indicial equation of $L^\delta_{c,v}$,
    (v) For each (up to conjugation over C) root e of this indicial equation, do :
      - Set $d := -e$ and add $(c, v, d + \mathbb{Z})$ to $\mathcal{A}$.
4. Return $\mathcal{A}$.

Remark 9. *For efficiency reasons, step* **3a** *is implemented slightly differently : Take $L_v := L_{\circledS}(\tau - x^v)$ and let $\tilde{L} = \sum a_{i,j} x^{-i} \delta^j = \sum a_{i,j} t^i \delta^j$ be the operator obtained from L_v by substituting $\tau = \delta + c$. We do not actually calculate $\tilde{L}$ completely, rather, we compute only those terms $a_{i,j} t^i \delta^j$ for which $i + j = v_1(\tilde{L})$. Then sum $a_{i,j}\, c^j\, (-1)^j\, e\, (e+1) \cdots (e+j-1)$ over those i, j to get the indicial equation. The factor c^j corrects for the fact that we did not include c in the computation of L_v for better efficiency (this helps*

in particular when c involves an algebraic extension). This way, the indicial equation generally takes very little CPU time to calculate.

Example 8. *We continue with example 7. The output of Algorithm* τPolygon *was* $\{[1, x^2 + 5\,x - 1], [-2, 10\,x - 8]\}$. *The first root we consider is* $c = \alpha$ *where* $\alpha = \mathrm{RootOf}(x^2 + 5\,x - 1)$. *Step* **3a** *then leads to one possibility for* $-d$ *namely* $-5/29\,\alpha + 89/29$. *The second possible c is* $4/5$ *which leads to* $-d = 0$. *Consequently, we have :*

$$\overline{g}_\infty(L) = \{(\alpha, -1, 5/29\,\alpha - 89/29),\ (4/5,\ 2,\ 0)\}.$$

Definition 37. *The* roots-number *of* $(c, v, d + \mathbb{Z}) \in \overline{g}_\infty(L)$ *is the number of distinct roots of the indicial equation of* $L^\delta_{c,v}$ *in* $e + \mathbb{Z}$ *where* $e = -d$. *The* formal-sol-dimension *of* $(c, v, d + \mathbb{Z})$ *is the dimension of the formal solutions of* $L^\delta_{c,v}$ *in* $t^e\,\overline{C}((t))$.

The formal-sol-dimension is less than or equal to the roots-number. Each of these two numbers can be used (see Proposition 8) to bound the degree of an extension of the field $C(c, d)$ over which possible hypergeometric solutions may be defined. In this paper we use the roots-number since it is less work to implement. However, instead one may also use the formal-sol-dimension which leads to a better bound. The same issue for the differential case is discussed in the comments on logarithms in [CH04, Section 7.1].

3.6 Representing the finite singularities

Let $L = a_n\,\tau^n + \cdots + a_0\,\tau^0$ where the a_i are polynomials in x with coefficients in a field C of characteristic zero. Suppose that C is the field of definition of L. In our algorithm we will use algebraic extensions C' of C but not splitting fields. Hence, we can not work with all roots of $a_0(x)\,a_n(x - n)$ simultaneously. This restriction is necessary because the construction of a splitting field can be prohibitively costly. This does not imply that there are any roots of $a_0(x)\,a_n(x - n)$ that we can not compute with, we can compute with each root, but we should compute with only one root at a time. This also means that throughout the computation we always need to keep track of what the base field C is and what the current extension field C' is, in particular, when factoring a polynomial it is essential (for efficiency as well as correctness of the algorithm) to specify the correct field. The algorithm SingOver$C$ below is essentially trivial, but we include it to emphasize this field issue. First another definition :

Definition 38. *Let* $f(x)$, $g(x) \in \overline{C}[x]$. *If there exists an integer* m *such that* $f(x) = g(x + m)$ *then* $f(x)$ *and* $g(x)$ *are said* shift equivalent.

Algorithm SingOver$C$ **:**

Input : $L \in C(x)[\tau]$ and the field C.
Output : the set of finite singularities of L over C, up to conjugation over C.
1. Sing $:= \emptyset$.
2. Let $\mathcal{S}$ be the set of monic irreducible factors of $a_n(x)\,a_0(x)$ in $C[x]$.

3. Now take a subset $\mathcal{T} \subseteq \mathcal{S}$ such that precisely one polynomial has been chosen in each shift equivalence class.
4. For all $P \in \mathcal{T}$, do :
  4a. Take a root x_P of P (see comments below),
  4b. Sing := Sing $\cup \{[P, x_P]\}$.
5. Return Sing.

The output Sing is given as couples $[P, x_P]$ where $P \in C[x]$ is irreducible over C and $x_P = \mathrm{RootOf}(P, x)$, which denotes the image of x in $C' = C[x]/(P)$. So the set Sing represents all finite singularities "up to conjugation over C" (see [CH04, 5.1] for details). Note that no calculation takes place in Step **4a**. Instead, this step involves the construction of a data structure : one needs to construct a field C' generated by one root x_P of the polynomial P.

After the set Sing has been computed, our base field is still C in many places in the algorithm. An exception is of course when we compute with a singularity $p = x_P + \mathbb{Z}$ in which case the base field is $C' = C(x_P)$.

Given Sing, if we want to determine the singularities over a field extension of C then we take each $[P, x_P]$ in Sing, we factor P over the field extension, and replace $[P, x_P]$ by the set of all $[P', x_{P'}]$ for all irreducible factors P' of P.

3.7 Computing e-solutions

Definition 39. *Let C be a field of characteristic zero. The* field of definition of a hypergeometric term $u(x)$ over C *is the smallest field $C' \supseteq C$ for which $r(x) \in C'(x)$ where $r(x) = u(x+1)/u(x)$. The* cvd-field of u *is the field $C(c, d)$ where c, d are as in Equation (3.4). For $L \in C(x)[\tau]$, the* cvd-fields of L *are the $C(c, d)$ for $(c, v, d + \mathbb{Z}) \in \overline{g}_\infty(L)$.*

The cvd-field of a hypergeometric term u is a subfield of the field of definition of u.

Definition 40. *Let C be a field of characteristic zero and $L \in C(x)[\tau]$. An* easy hypergeometric solution*, or* e-solution *for short, is a hypergeometric solution of L defined over its cvd-field.*

Definition 41. *For $r \in \overline{C}(x)$, let* $\mathrm{hyp}(r) \in \mathcal{V}$ *denote a non-zero solution of $\tau - r$.*

Note that $\mathrm{hyp}(r)$ is not uniquely defined, but only defined up to a constant factor. However, that will not play a role in this paper.

Example 9. *Let $L \in \mathbb{Q}(x)[\tau]$ and assume that L has both $u_1 := \mathrm{hyp}(x^2 + \sqrt{2})$ and $u_2 := \mathrm{hyp}(x + \sqrt{2})$ as hypergeometric solutions. By definition, u_1 is a hypergeometric solution of $\tau - (x^2 + \sqrt{2})$ so its local type at infinity is $(1, -2, \mathbb{Z})$ since $x^2 + \sqrt{2} = 1\,t^{-2}\,(1 + O(t^2))$ with $t = 1/x$, and so its cvd-field is $\mathbb{Q}$: it follows that this is not an e-solution. Now, $x + \sqrt{2} = 1\,t^{-1}\,(1 + \sqrt{2}\,t^1 + O(t^2))$ thus the local type at infinity of u_2 is $(1, -1, \sqrt{2} + \mathbb{Z})$ and its cvd-field is $\mathbb{Q}(\sqrt{2})$ so u_2 is an e-solution of L.*

We will give an algorithm, called ESols, that computes all e-solutions having a given local type $(c, v, d+\mathbb{Z})$ at infinity. In this algorithm, two main parts can be distinguished : an *initialization* phase and a *search* phase. In the sequel, we will need to use the full algorithm (in HypSols in Section 9) as well as the search phase alone (in HSol in Section 8) : for this reason, we split up the algorithm into two procedures.

The first one, called Search, will only be called once the finite singularities $[P_1, x_1], \ldots, [P_s, x_s]$ over C and the $\overline{g}_{x_i}(L)$ have already been calculated. Moreover, this procedure searches for e-solutions (that are defined over $C(c, d)$) so that the P_i have to be factored over $C(c, d)$: in the algorithm, we denote $P_i = P_{i,1} \cdots P_{i,m_i}$ this factorization and the $P_{i,j}$ (irreducible over $C(c, d)$) are given as input. The field C' in the input of Algorithm Search will be $C(c, d)$ if Search is called from Algorithm ESols, or an extension of $C(c, d)$ when called from Algorithm HSol in the next section.

Algorithm Search **:**
Input : $L \in C(x)[\tau]$, the field C', $(c, v, d + \mathbb{Z}) \in \overline{g}_\infty(L)$, the $\overline{g}_{x_i}(L)$ and the $P_{i,j}$.
Output : Sol, a basis of all e-solutions of L with local type $(c, v, d + \mathbb{Z})$ at ∞.
1. Let Sol $:= \emptyset$.
2. For each combination $(l_{1,1}, \ldots, l_{1,m_1}, \ldots, l_{s,m_s})$ with $l_{i,j} \in \overline{g}_{x_i}(L)$ that satisfies the two Fuchs' relations (Equation (3.6)), do :
2a. Let $r := c \prod_{i,j} P_{i,j}^{l_{i,j}}$,
2b. Compute $R \in C'(x)$ (see comments below) and let $\overline{r} := r \, \frac{\tau(R)}{R}$,
2c. Compute $\tilde{L} := L_{\circledS}(\tau - 1/\overline{r})$,
2d. Compute a basis $Q_1, \ldots, Q_w$ of polynomial solutions in $C'[x]$ for $\tilde{L}$,
2e. If $w > 0$, then add the $\mathrm{hyp}(\overline{r})\, Q_i$ to Sol.
3. Return Sol.

Regarding Step **2b**, we have to compute the rational solutions of the operator $L_{\circledS}(\tau - 1/r)$. The computation in Section 4 produces $\overline{g}_{x_i}(L)$ as well as the numbers $B^l(q)$ and $B^r(q)$ in Equation (3.10). Given these $B^l(q), B^r(q)$ for L, it is trivial to deduce the numbers $B^l(q), B^r(q)$ for the operator $L_{\circledS}(\tau - 1/r)$. The numbers $B^l(q)$ and $B^r(q)$ were used in [Ho98] to compute a denominator bound for rational solutions, more precisely, they were used to compute a rational function R such that all rational solutions are polynomials times R (this R is a product of powers of shifts of the $P_{i,j}$). Thus, rational solutions of $L_{\circledS}(\tau - 1/r)$ correspond to polynomial solutions of the operator $\tilde{L}$ in Step **2c**. Given $(c, v, d + \mathbb{Z})$ and the numbers $B^l(q), B^r(q)$, it is easy to calculate the degree bound for the polynomial solutions of $\tilde{L}$ *before* computing R and $\tilde{L}$. Our implementation does this so that computing $\tilde{L}$ can be avoided whenever the degree bound is a negative integer.

To use this procedure in the recursive algorithm HSol in Section 8, we need to allow one extra input, namely an option called "just one" ; if this option is given then we only need to return just one (if it exists) e-solution but not a totally arbitrary one. We return a $\mathrm{hyp}(\overline{r})\, Q$ with Q of minimal possible degree.

Now we give the complete ESols Algorithm which after an initialization of the local informations at finite singularities calls Search.

Algorithm ESols **:**
Input : $L \in C(x)[\tau]$, the field C and $(c, v, d+\mathbb{Z}) \in \overline{g}_\infty(L)$.
Output : Sol, a basis of all e-solutions of L with local type $(c, v, d+\mathbb{Z})$ at ∞.
1. Let $\mathcal{S} :=$ SingOverC(L, C) and $s :=$ Card$(\mathcal{S})$.
2. For $[P_i, x_i] \in \mathcal{S}$, compute $\overline{g}_{x_i}(L)$.
3. Let $C' := C(c, d)$ be the cvd-field.
4. For $i \in \{1, \ldots, s\}$, factor P_i over C' : let $P_i := P_{i,1} \cdots P_{i,m_i}$.
5. Sol := Search(L,C',$(c, v, d+\mathbb{Z})$,$\{\overline{g}_{x_i}(L)\}_i$,$\{P_{i,j}\}_{i,j}$).
6. Return Sol.

We store the result from steps 1 and 2 so that the computed information can be re-used later without repeating this computation, for example, we can re-use this information if ESols is called again with the same L but a new $(c, v, d+\mathbb{Z})$.

3.8 Computing a hard solution

Let C be a field of characteristic zero and $L \in C(x)[\tau]$. In this paper we aim to find a basis of all hypergeometric solutions of L so we may have to search for hypergeometric solutions defined over algebraic extensions of the cvd-fields of L. We first bound the degrees of such extensions.

As in the differential case (see [CH04, Section 6.1]), we define the notion *the minimal algebraic degree* :

Definition 42. *Let C be a field of characteristic zero and u be a hypergeometric term. Let C' be the field of definition of u over C (see Definition 39). Then the* algebraic degree *of u over C is $[C' : C]$. We say that u is of* minimal algebraic degree m over C *if $[C' : C] = m$ and there exist no hypergeometric term v, of the same type as u, having smaller algebraic degree over C.*

Then, adapting the exposition in [CH04, Section 6.1] from the differential case, we can prove the following ([CH04, Lemma 6.3]) :

Lemma 10. *Let C be a field of characteristic zero and $L \in C(x)[\tau]$. If u is a hypergeometric solution of L defined over some algebraic extension C' of C, then its conjugates over C are also solutions of L.*
Given L and u, we can compute $u_1, \ldots, u_k$ of minimal algebraic degree over C that form a basis of all hypergeometric solutions of L of the same type as u.
If u is of minimal algebraic degree over C, then the distinct conjugates of u over C are of distinct type and hence linearly independent.

This implies that a hypergeometric solution of L of minimal algebraic degree m corresponds to m linearly independent hypergeometric solutions of L. In Algorithm HypSols in Section 9, we only return hypergeometric solutions up to conjugation over

C. Therefore to make it easier to count the number of linearly independent solutions represented by this output, we want HypSols to only return hypergeometric solutions of minimal algebraic degree.

Proposition 8. *Let $L \in C(x)[\tau]$ where C is a field of characteristic zero and let $(c, v, d+\mathbb{Z}) \in \overline{g}_\infty(L)$. Let $B_1(c, v, d)$ be the roots-number of $(c, v, d+\mathbb{Z})$ (see Definition 37). If u is a hypergeometric solution of minimal algebraic degree m over the cvd-field of u with local type $(c, v, d+\mathbb{Z})$ at infinity, then $m \leqslant B_1(c, v, d)$.*

Proof. From Lemma 10, the conjugates of u form m linearly independent hypergeometric solutions of L, all of which have local type $(c, n, d+\mathbb{Z})$ at infinity. Now (see Section 5), these hypergeometric solutions correspond to formal solutions at infinity of $L^\delta_{c,v}$ that can be written $t^{-d}\, F_i(t)$, $i \in \{1, \ldots, m\}$, where $F_i(t) \in \overline{C}[[t]]$. After Gaussian elimination on the vectors of coefficients of the $F_i(t)$, we obtain m formal solutions $t^{-d}\, \tilde{F}_i(t)$ of $L^\delta_{c,v}$ with distinct exponents. Hence, we can conclude $m \leqslant B_1(c, v, d)$ by Definition 37.□

Let $L \in C(x)[\tau]$ where C is some field of characteristic zero and suppose that L has no e-solutions of local type $(c, v, d+\mathbb{Z})$ at infinity. We want to compute (if it exists) a solution with local type $(c, v, d+\mathbb{Z})$ at infinity defined over an algebraic extension of its cvd-field. To improve the efficiency of the final algorithm HypSols, we will first compute the e-solutions and reduce the order before entering the "hard cases". In these cases, where we must introduce algebraic extensions of the cvd-fields of L, we want to stop as soon as a solution is found since it also leads to a reduction of order. Moreover as this algorithm is only run once the e-solutions have been computed, the finite singularities $\text{Sing} = \{[P_1, x_1], \ldots, [P_s, x_s]\}$ over C and the $\overline{g}_{x_i}(L)$ are already known. Algorithm HypSols calls Algorithm HSol with arguments : L, a cvd-field $C(c, d)$ of L, $(c, v, d+\mathbb{Z})$, a bound B, the set Sing, and the empty set for $\mathcal{F}$. So the field that is denoted as C in Algorithm HSol is initially a cvd-field $C(c, d)$, and so in Algorithm HSol we always have c, $d \in C$.

Algorithm HSol **:**
Input : $L \in C(x)[\tau]$, a field C, $(c, v, d+\mathbb{Z}) \in \overline{g}_\infty(L)$, $B \in \mathbb{N}$, a set Sing and a set $\mathcal{F}$.
Output : a hypergeometric solution of L with local type $(c, v, d+\mathbb{Z})$ at ∞ or $\emptyset$.
1. If $B = 0$, then return $\emptyset$ and stop.
2. If option "no extension", then $\mathcal{S} :=$ Sing, otherwise :
2a. For all $[P_i, x_i] \in$ Sing, do :
2a1. If $[P_i, x_i] \in \mathcal{F}$, then $P_{i,1} := P_i$, otherwise factor P_i over C, $P_i = P_{i,1} \cdots P_{i,m_i}$,
2b. If Search(L,C,$(c, v, d+\mathbb{Z})$,$\{\overline{g}_{x_i}(L)\}_i$,$\{P_{i,j}\}_{i,j}$, option "just one") $\neq \emptyset$, then return it and stop.
2c. If $B = 1$, then return $\emptyset$ and stop.
2d. Let $\mathcal{S}$ be the set of all $[P_{i,j}, x_{i,j}]$ and let $\overline{g}_{x_{i,j}}(L) := \overline{g}_{x_i}(L)$ for all i, j.
3. If $\mathcal{S} \setminus \mathcal{F}$ contains a $[P_i, x_i]$ such that x_i is an essential singularity and $[C(x_i) : C] > 1$, then :
3a. If HSol(L,C,$(c, v, d+\mathbb{Z})$,B,$\mathcal{S}$,$\mathcal{F} \cup \{[P_i, x_i]\}$, option "no extension") $\neq \emptyset$, then return it and stop.
3b : If HSol(L, $C(x_i)$, $(c, v, d+\mathbb{Z})$, $\lfloor B \frac{\lfloor \deg(P_i)/2 \rfloor}{\deg(P_i)} \rfloor$,$\mathcal{S}$,$\mathcal{F}$) $\neq \emptyset$, then return it and stop.
4 : Return $\emptyset$.

We must now prove that this algorithm satisfies the following :

Algorithm specification : if there exists a hypergeometric solution u that satisfies the information encoded by $\mathcal{F}$ (meaning that none of the minimal polynomials in $\mathcal{F}$ becomes reducible over the field of definition of u) and that has minimal algebraic degree $\leqslant B$ over C, then HSol will return some hypergeometric solution of L (but not necessarily of degree $\leqslant B$ over C).

Proof. Suppose that a hypergeometric solution u exists, satisfies the information encodes by $\mathcal{F}$ and has degree $\leqslant B$ over C. Let C' be the field of definition of this u over C. Consider the $[P_i, x_i] \in \mathcal{S}$ for which x_i is an essential singularity. If none of these P_i become reducible over C', then there are no more combinations over C' than there are over C. Thus, if there are no hypergeometric solutions defined over C but there is a hypergeometric solution u defined over C' then at least one P_i must factor over C'. In Step **3** we pick a $[P_i, x_i]$, which may have been a wrong choice (P_i stays irreducible over C'), in which case Step **3a** will work, but it may also have been a right choice (meaning that P_i is reducible over C'), in which case Step **3b** will work. We then have to prove that in this case the bound given is correct. Suppose that P_i has degree d_i. P_i is reducible over C' and thus one of its irreducible factors will have degree $d_i' \leqslant \lfloor d_i/2 \rfloor$. Note that $\lfloor d_i/2 \rfloor / d_i = 1/2$ when d_i is even, and $\lfloor d_i/2 \rfloor / d_i < 1/2$ when d_i is odd. By abuse of notation, we also denote x_i as a root of that factor. So then $[C'(x_i) : C'] = d_i'$, $[C' : C] \leqslant B$, so $[C'(x_i) : C] \leqslant d_i' B$. Now $C(x_i)$ is a field between $C'(x_i)$ and C having degree d_i over C, so $[C'(x_i) : C(x_i)] = [C'(x_i) : C]/d_i \leqslant d_i' B/d_i \leqslant \lfloor d_i/2 \rfloor B/d_i$. Now u is defined over C', hence also over $C'(x_i)$, which is an extension of degree $\leqslant \lfloor d_i/2 \rfloor B/d_i$ over our new field $C(x_i)$. Hence the degree bound used in Step **3b** is correct.□

3.9 An algorithm to find all hypergeometric solutions

In this section, we present an algorithm to compute a basis of all hypergeometric solutions (up to conjugation) of a difference operator L with coefficients in $C(x)$ where C is a field of characteristic zero. Our algorithm uses the same strategy and satisfies the same specifications as Algorithm ExpSols in [CH04, Section 7].

Algorithm HypSols **:**
Input : a linear difference operator $L \in C(x)[\tau]$ and the field C.
Output : Sol, a basis of hypergeometric solution of L up to conjugation over C.
1 : Sol $:= \emptyset$.
2 : $\mathcal{A} :=$ LocalTypeAt$\infty(L,C)$.
3 : For $(c, v, d+\mathbb{Z}) \in \mathcal{A}$, do :
3a : $\mathrm{V}_{(\mathrm{c,v,d})} :=$ ESols$(L,C,(c, v, d+\mathbb{Z}))$,
3b : Sol $:=$ Sol $\cup\, \mathrm{V}_{(\mathrm{c,v,d})}$,
3c : Let $B_{(c,v,d)}$ be a bound for the number of independent hypergeometric solutions with local type $(c, v, d+\mathbb{Z})$ at infinity,
3d : $B_{(c,v,d)} := B_{(c,v,d)}$ - Card$(\mathrm{V}_{(\mathrm{c,v,d})})$,
3e : If $B_{(c,v,d)} \leqslant 1$, then $\mathcal{A} := \mathcal{A} \setminus \{(c, v, d+\mathbb{Z})\}$.
4 : If $\mathcal{A} = \emptyset$, then return Sol and stop.
5 : If Sol $\neq \emptyset$, then
5a : Write $L = \tilde{L}$ LCLM$(\tau - \tau(s)/s$ and conjugates over $C \,|\, s \in$ Sol$)$, and run HypSols$(\tilde{L},C)$,
5b : Of the solutions in **5a**, keep only the types not defined over their cvd-field,
5c : For each remaining type represented by t, do :
– Compute a basis $R_1, \ldots, R_s$ of rational solutions of $L_{\circledS}(\tau - 1/t)$,
– Sol $:=$ Sol $\cup\, \{\mathrm{hyp}(t)R_1, \ldots, \mathrm{hyp}(t)R_s\}$.
5d : Return Sol and stop.
6 : If the order is 2 then go to **10**.
7 : Make L monic and let L^* be the adjoint of L.
8 : Let S be the union of ESols$(L^*,C,(c, v, d+\mathbb{Z}))$ for all $(c, v, d+\mathbb{Z}) \in \overline{g}_\infty(L^*)$.
9 : If $S \neq \emptyset$, then
9a : Write $L =$ LCLM$(\tau - \tau(s)/s$ and conjugates over $C \,|\, s \in S)^*\, \tilde{L}$,
9b : Return HypSols$(\tilde{L},C)$ and stop.
10 : For $(c, v, d+\mathbb{Z}) \in \mathcal{A}$, do :
10a : HSol$(L,C(c, d),(c, v, d+\mathbb{Z}),B_{(c,v,d)},\mathrm{Sing},\emptyset)$,
10b : If it finds a solution, then
– Optimize the solution found, denote it as u, and let $r = \tau(u)/u$,
– Compute a basis $R_1, \ldots, R_s$ of rational solutions of $L_{\circledS}(\tau - 1/r)$,
– Sol $:= \{u\, R_1, \ldots, u\, R_s\}$,
– Write $L = \tilde{L}$ LCLM$(\tau - \tau(s)/s$ and conjugates over $C \,|\, s \in$ Sol$)$,
– Remove recursively from $\tilde{L}$ the solutions of the same type as conjugates of u,
– Run HypSols$(\tilde{L},C)$ and of its output, keep only types that are not defined over their cvd-field,
– For each remaining type represented by t, do :
Compute a basis $R_1, \ldots, R_s$ of rational solutions of $L_{\circledS}(\tau - 1/t)$,
Sol $:=$ Sol $\cup\, \{\mathrm{hyp}(t)R_1, \ldots, \mathrm{hyp}(t)R_s\}$.
– Return Sol and stop.
11 : Return $\emptyset$ and stop.

We explain or comment some points in the algorithm. Some remarks have already been made in [CH04].

- The bound $B_{(c,v,d)}$ used in Step **3c** is usually the $B_1(c, v, d)$ of Proposition 8. However, other bounds can be used : the formal-sol-dimension of $(c, v, d + \mathbb{Z})$ (see Definition 37) or the bound coming from computations modulo a prime p (see Proposition 10).
- When we have applied recursion on a left factor of L (Steps **5** or **10b**), the hypergeometric solutions found are in general not hypergeometric solutions of L. We will only use the types of the hypergeometric solutions of this left factor, not the hypergeometric solutions themselves. We skip types defined over their cvd-field because all such hypergeometric solutions have already been found.
- We will apply HSol only when all easy factors have been removed both on the left and on the right. Removal of easy factors, left or right, does not cause solutions to be lost because we only do this after all easy solutions have already been computed and stored in the set Sol, in Steps **1**,**2**,**3** of the algorithm. To find the easy left factors, in Step **8**, we apply ESols to the adjoint operator L^* (see for example [We01, Definition 3.2.1]) to find the easy hypergeometric solutions of L^* which correspond to easy left factors of L. Then we apply recursion on the remaining right hand factor.
- When we reach the point (Step **10**) where there is nothing left to do than entering the "hard case", we try to find a solution over an extension of a cvd-field with HSol. An *a priori* first bound is $B_1(c, v, d)$ (see Proposition 8). The variable Sing in Step **10a** is calculated during the first call to ESols in Step **3a**.
- Let u be a hypergeometric solution. We can write $\tau(u)/u = P_1/P_2$ where P_1 and P_2 are polynomials with gcd 1 and P_2 is monic. The field of definition of u is then the field generated over C by the coefficients of P_1 and P_2. In step **10b**, by "optimizing the solution" we mean two things : 1). Using this solution u to find a solution of minimal algebraic degree over C. And 2). Making sure that the field that the algorithm gives for u (this field contains the field of definition of u) is actually equal to the field of definition of u. Both 1) and 2) are important. We want u to be of minimal algebraic degree so that we know that its conjugates are linearly independent. But the way we count the number of conjugates of u is not by looking at u, but by looking at the field given for u. The field provided for u by Algorithm HSol contains, but need not be not equal to, the field of definition of u. So to optimize u, we take the field given for u, and then determine the subfield generated over C by the coefficients of P_1 and P_2. Then we find defining equations (*i.e.*, new RootOf's) for this subfield, and use them to rewrite the coefficients of P_1 and P_2. This then takes care of 2). To do 1), we could use the approach in Lemma 10, however, this is not necessary because the special choice that Algorithm ESols makes when the option "just one" is given causes 1) to be automatically satisfied. The polynomial Q is of minimal degree with this option given, which leads to uniqueness of $\tau(u)/u$ which in turn causes u to already be of minimal algebraic degree over C.
- When we "remove recursively from $\tilde{L}$ the solutions of the same type as u" (Step **10b**), we also remove solutions whose type is conjugated over C to the type of u. We will remove such solutions recursively because if we do not then it would be non-trivial to ensure that the "same" solution is not returned more than once. So before we call HypSols on the remaining left factor $\tilde{L}$ of L, we first remove from $\tilde{L}$ all hypergeometric solutions of the same type as (conjugates of) u, and we keep repeating this until $\tilde{L}$ no

longer has solutions of this type. A recursive procedure to achieve that is the following : compute the rational solutions R_i of $\tilde{L}_{\circledS}(\tau - 1/r)$ where $r = \tau(u)/u$ is the certificate of u. Each R_i gives a first order right hand factor $\tau - r_i$ of $\tilde{L}$ where $r_i = r\tau(R_i)/R_i$. If there are none, then return $\tilde{L}$ else write $\tilde{L} = \tilde{L}_1$ LCLM($\tau - r_1, \ldots, \tau - r_s$, "and conjugates over C") and apply recursion on $\tilde{L}_1$.

3.9.1 Some possible improvements

We shall give two possible improvements that can be added to Algorithm HypSols. The first one is included in the Maple implementation.

Maps between $V_{p,l}(L)$ and $V_{p,r}(L)$, and possible degrees of field extensions : we show how the ranks of the maps between left and right solution spaces $V_{p,l}(L)$ and $V_{p,r}(L)$ at finite singularities p can be used to discard certain cases during the search phase of the algorithm.
Let p be a finite singularity of L. Let $g_{p,r} = g_{p,r}(L)$ be the smallest and $g_{p,l} = g_{p,l}(L)$ the largest element of $\overline{g}_p(L)$. In [Ho99, Section 4.2], the author shows how to construct two maps $E_{p,r} : V_{p,l}(L) \to V_{p,r}(L)$ and $E_{p,l} : V_{p,r}(L) \to V_{p,l}(L)$ by defining their action on explicit basis. In our implementation, we only use the rank of these maps. If rk_r is the rank of $E_{p,r}$, and rk_l is the rank of $E_{p,l}$, then this gives us the following information :

- there can be at most rk_r linearly independent hypergeometric solutions u with $g_p(u) = g_{p,r}$ (minimal valuation growth),
- at most $n - \mathrm{rk}_r$ with $g_p(u) > g_{p,r}$ (more than minimal valuation growth),
- at most rk_l with $g_p(u) = g_{p,l}$ (maximal valuation growth), and,
- at most $n - \mathrm{rk}_l$ with $g_p(u) < g_{p,l}$ (less than maximal valuation growth).

This can be exploited in several ways :

1. Suppose that there can be at most 2 independent solutions with less than maximal valuation growth, and suppose that the algorithm already found 2 such solutions. Then from that moment on, we no longer need to consider the case $g_p(u) < g_{p,l}$, and so we may assume $g_p(u) = g_{p,l}$.
2. Or suppose that there can be at most 1 linearly independent solution with minimal valuation growth. Then, if there exists such a solution u with $g_p(u)$ minimal, and if $p = q + \mathbb{Z}$ with $q \in C$, then the certificate $r(x) = u(x+1)/u(x)$ of u must be an element of $C(x)$. The reason for that is that if $r(x)$ was defined over an extension C' of C, then by taking conjugates of $r(x)$ over C' we would find more than 1 linearly independent solution with minimal valuation growth at p. More generally, suppose that there are at most s_1 independent solutions with maximal valuation growth, but s_2 have already been found. So there can remain at most $s_1 - s_2$ independent hypergeometric solutions $u(x)$ with maximal valuation growth, and this means that for computing the certificate $r(x) = u(x+1)/u(x)$ of such a solution, it is not necessary to consider field extensions of degree more than $s_1 - s_2$.

These results are generalizations of [Ho99, Theorems 2 and 3] and can be proven in a similar way.

Sorting the local types at infinity : it can be useful to sort the local types at infinity in such a way that we compute the solutions that have the smallest fields of definition first. The reason is that each time a solution is found, the number of combinations to be considered may be reduced following the items above, so we want to compute first those solutions that are the least expensive to calculate.

3.10 Modular improvements

We shall propose a number of improvements we can add to Algorithm HypSols using p-curvature computations. This section has its own theoretical interest since, as it is done in [CH04] for the differential case, we give links between the local information in characteristic zero and the roots of the characteristic polynomial of the p-curvature matrix.

3.10.1 Difference operators in characteristic p

In this part, most results are stated without proof since they are just adaptations of results in [CH04, 1.2] from the differential to the difference case. A classification (similar to that for the differential case) of difference modules in characteristic p can be found in [PS97, Chapter 5].

Let $\overline{\mathbb{F}}_p$ denote the algebraic closure of the finite field $\mathbb{F}_p$. We define the ring $\overline{\mathbb{F}}_p(x)[\tau]$ of difference operators in the same way as in characteristic zero. The main difference is that the constant field is then $\overline{\mathbb{F}}_p(x^p - x)$ (see [PS97, Chapter 5] or [GZ03, Theorem 3.1]). To simplify the notations, we set :

$$\lambda := x^p - x.$$

The central tool to study recurrence relations in characteristic p is the p-curvature; we give a definition of the *p-curvature* when the equation is written in matrix form $Y(x+1) = A(x)\,Y(x)$ with $A(x) \in \mathcal{M}_n(\overline{\mathbb{F}}_p(x))$ and for more details we refer to [PS97, Chapter 5] and references therein. Note that a difference operator $L \in \overline{\mathbb{F}}_p(x)[\tau]$ is naturally associated to a linear difference equation in matrix form; take for $A(x)$ the companion matrix of L.

Definition 43. *Let $Y(x+1) = A(x)\,Y(x)$ with $A(x) \in \mathcal{M}_n(\overline{\mathbb{F}}_p(x))$ be a linear difference equation. Its* p-curvature *is defined as the product of matrices $A(x+p-1)\cdots A(x+1)\,A(x)$.*

Consider the following group homomorphisms :

$$\tau_1 : \overline{\mathbb{F}}_p(x)^* \to \overline{\mathbb{F}}_p(\lambda)^*,\ \ u \mapsto u(x+p-1)\cdots u(x+1)\,u(x),$$

and

$$\tau_2 : \overline{\mathbb{F}}_p \to \overline{\mathbb{F}}_p,\ a \mapsto a^p - a,$$

which are related by Equation (3.12) below.

Lemma 11. (i) : *The map* $\tau_1 : \overline{\mathbb{F}}_p(x)^* \to \overline{\mathbb{F}}_p(\lambda)^*$ *is a surjective multiplicative map with kernel* $\{\tau(u)/u \mid u \in \overline{\mathbb{F}}_p(x)^*\}$.
(ii) : *The map* τ_2 *is a surjective additive map with kernel* $\mathbb{F}_p$.

Proof. **(ii)** is obvious, and **(i)** follows from **(ii)** and Equation (3.12) below (one can also follow a similar result for the differential case in [Pu95, Lemma 1.4.2]).□

For $L \in \overline{\mathbb{F}}_p(x)[\tau]$, let $\chi_p(L)$ denote the characteristic polynomial of the p-curvature of L. As we are only interested in first order factors, the only modular information that we can use are the roots in $\overline{\mathbb{F}}_p(\lambda)$ (with multiplicities) of $\chi_p(L)$. We denote $\mathcal{R}(\chi_p(L))$ as the set of roots of $\chi_p(L)$ in $\overline{\mathbb{F}}_p(\lambda)$.

Lemma 12. *Let* $L \in \overline{\mathbb{F}}_p(x)[\tau]$.
(i) : *If* $L = L_1 L_2$, *then* $\chi_p(L) = \chi_p(L_1)\chi_p(L_2)$ *with* $\deg(\chi_p(L_i)) = \operatorname{order}(L_i)$.
(ii) : *The map from the set of monic first order right hand factors of* L *in* $\overline{\mathbb{F}}_p(x)[\tau]$ *to* $\mathcal{R}(\chi_p(L))$ *defined by* $\tau - r \mapsto \tau_1(r)$ *is well defined and surjective.*

As in the differential case, the key ingredient to use modular information for computing hypergeometric solutions of $L \in \overline{C}(x)[\tau]$ is the known fact that, with few assumptions on the prime p to be chosen, a factorization of L in characteristic zero can be reduced mod p.

Definition 44. *For some object* f *in characteristic zero, we note* $f[p]$ *its reduction modulo* p *in the sense of [CH04, Sections 1.3 and 7.1].*

The following proposition and its corollary can be proven as [CH04, Proposition 1.15] and [CH04, Corollary 1.16].

Proposition 9. *Let* C *be a field of characteristic zero, let* $L \in C(x)[\tau]$ *and let* p *be such that* L *can be reduced mod* p. *If* $L = L_1 L_2$ *then, after possibly replacing* L_1 *and* L_2 *by* cL_1 *and* $\frac{1}{c}L_2$ *for some constant* $c \in C$, *we have* $L[p] = L_1[p]\ L_2[p]$.

Corollary 3. *Let* C *be a field of characteristic zero, let* $L \in C(x)[\tau]$ *and let* p *be a prime such that* L *can be reduced mod* p. *Assume further that the order* n *of* L *does not drop after the reduction.*
(i) : *If* $\chi_p(L)$ *is irreducible over* $\overline{\mathbb{F}}_p(\lambda)$, *then* L *is irreducible over* $\overline{C}(x)$.
(ii) : *If* $\chi_p(L)$ *has no roots in* $\overline{\mathbb{F}}_p(\lambda)$, *then* L *has no hypergeometric solutions.*

3.10.2 Local types at infinity and $\mathcal{R}(\chi_p(L))$

Definition 45. *If* a *is an object in characteristic zero, then for* $i \in \{1, 2\}$, *we set* $\tau_i(a) := \tau_i(a[p])$.

If $d \in \overline{C}$ and $d[p]$ is defined then $\tau_2(d + \mathbb{Z})$ is defined since $\tau_2(i) = 0$ for $i \in \mathbb{Z}$.

Lemma 13. *Let* $r(x) = c\frac{\tau(Q)}{Q}\prod_{i=1}^{m}(x-x_i)^{e_i} \in \overline{C}(x)$ *with* $c, x_i \in \overline{C}$, $e_i \in \mathbb{Z}$ *and* $Q \in \overline{C}[x]$. *Assume further that the* x_i, c, Q *can be reduced* mod p *and that* $Q[p] \neq 0$. *Then*

$$\tau_1(r(x)) = c^p \prod_i (\lambda - \tau_2(x_i))^{e_i}. \tag{3.12}$$

Furthermore, if $r(x)$ *has expansion* $c\,t^v\,(1 + d\,t + \mathcal{O}(t^2))$ *at infinity, then*

$$\tau_1(r(x)) = c^p\,t_\lambda^v\,(1 + \tau_2(d)\,t_\lambda + \mathcal{O}(t_\lambda^2)), \tag{3.13}$$

where

$$t_\lambda := \frac{1}{\lambda} = \frac{1}{x^p - x}.$$

Proof. We have $\tau_1(r(x)) = \tau_1(c\frac{\tau(Q)}{Q}\prod_i(x-x_i)^{e_i}) = \tau_1(c)\prod_{i=1}^{s}(\tau_1(x-x_i))^{e_i}$ since, from Lemma 11, τ_1 is multiplicative and $\tau(Q)/Q \in \ker(\tau_1)$. Now, clearly $\tau_1(c) = c^p$. Furthermore, we have $\tau_1(x) = (x+p-1)\cdots(x+1)\,x = x^p - x$ and substituting $x = x - x_i$ in this equality leads to $\tau_1(x-x_i) = (x-x_i)^p - (x-x_i) = \lambda - \tau_2(x_i)$. Equation (3.12) is then clear.
For (3.13), we only need to prove that the coefficient $\tau_2(d)$ of t_λ is correct. Let $\tilde{d}$ denote this coefficient. From (3.12), we have $\tilde{d} = -\sum_i e_i\tau_2(x_i)$. But we also know from Section 1 that $d = -\sum_i e_i x_i$ so that an easy verification leads to $\tau_2(d) = \tilde{d}$.□

Definition 46. *Let* $L = a_n\tau^n + \cdots + a_0\tau^0 \in C(x)[\tau]$ *for some field* C *of characteristic zero and assume that* a_n, $a_0 \neq 0$. *After multiplying* L *on the left by a suitable element of* $C(x)$, *we may assume the coefficients* a_i *are in* $C[x]$ *and* $\gcd(a_0, \ldots, a_n) = 1$.
We say that a prime p *is* a good prime for L *if the two following conditions hold :*
(C1) : *the coefficients of* L *can be reduced* mod p *and* $a_0[p] \neq 0$, $a_n[p] \neq 0$.
(C2) : *the finite singularities can be reduced* mod p.

Condition **(C1)** means that the order does not decrease after reduction mod p. Condition **(C2)** simply means that the leading coefficient of the polynomials a_0 and a_n do not reduce to zero mod p.

We give now a new result linking local types of hypergeometric solutions and eigenvalues of the p-curvature in $\overline{\mathbb{F}}_p(\lambda)$.

Theorem 2. *Let* C *be a field of characteristic zero, let* $L \in C(x)[\tau]$ *and let* p *be a good prime for* L. *Suppose that* $\tau - r$ *is a right hand factor of* L *with* $r \in \overline{C}(x)$. *Let* $s = \tau_1(r) \in \mathcal{R}(\chi_p(L))$. *Then there exist* $e_i \in \overline{g}_{x_i}(L)$ *at each finite singularity* x_i *of* L *and* $(c, v, d + \mathbb{Z}) \in \overline{g}_\infty(L)$ *such that* c, d *can be reduced mod* p *and*
(i) : $s = c^p\prod_i(\lambda - \tau_2(x_i))^{e_i}$,
(ii) : *the expansion at infinity of* s *is* $c^p\,t_\lambda^v\,(1 + \tau_2(d)\,t_\lambda + \mathcal{O}(t_\lambda^2))$.

Proof. One can prove (for example, adapt the proof of [CH04, Lemma 1.8] to the difference case) that if $\tau - r$ is a first order right hand factor of L then there exist a local

type $e_i \in \overline{g}_{x_i}(L)$ at each finite singularity x_i of L, a local type $(c, v, d+\mathbb{Z}) \in \overline{g}_\infty(L)$ at infinity and a rational function $R \in \overline{C}(x)$ such that :

$$r = c\,\frac{\tau(R)}{R}\prod_i (x-x_i)^{e_i} = c\,t^v\,(1+d\,t+\mathcal{O}(t^2)).$$

Since p satisfies conditions **(C1)** in Definition 46, r can be reduced mod p (see Proposition 9). Now from Condition **(C2)**, $\prod_i (x-x_i)^{e_i}$ can be reduced mod p and is further not zero mod p. Furthermore, R is defined modulo a multiplicative constant which we can choose in such a way that $R[p]$ is defined and not zero. Then $\tau(R)/R$ is also defined and not zero mod p. Consequently, c can be reduced mod p, and $c[p] \neq 0$, because c is the quotient of r and $\frac{\tau(R)}{R}\prod_i (x-x_i)^{e_i}$ which both reduce to something non-zero mod p. Then d can also be reduced mod p. Now the two formulas **(i)** and **(ii)** follows directly from Lemma 13.□

3.10.3 How to improve HypSols using the p-curvature ?

The first improvement we propose for HypSols is a bound unrelated to the one of Proposition 8. Consequently, in practice, we may use the minimum of all these bounds. This bound is, in some cases, used in the Maple implementation.

Proposition 10. *Let C be a field of characteristic zero, let $L \in C(x)[\partial]$ and let p be a good prime for L. Let $(c, v, d+\mathbb{Z}) \in \overline{g}_\infty(L)$. Let $B_2(c, v, d)$ be the number of roots s of $\chi_p(L)$ (counted with multiplicity) that have expansion $c^p\,t_\lambda^n\,(1+\tau_2(d)\,t_\lambda+\mathcal{O}(t_\lambda^2))$ at infinity. If u is a hypergeometric solution of minimal algebraic degree m over its cvd-field with local type $(c, v, d+\mathbb{Z})$ at infinity, then $m \leqslant B_2(c, v, d)$.*

Proof. Let C' be the cvd-field. Let $R \in C'(x)[\tau]$ be the right hand factor of L whose solutions are spanned by all hypergeometric solutions u of L that have local type $(c, v, d+\mathbb{Z})$ at infinity. Since there are at least m independent such solutions, the order o of R is at least m. All solutions of R have the same local type $(c, v, d+\mathbb{Z})$ at infinity. Now R factors as a product of first order factors $R = R_1 R_2 \cdots R_o$ because R has a basis of hypergeometric solutions. All R_i have the same local type at infinity, so for each R_i, the local type at infinity must be $(c, v, d+\mathbb{Z})$. Hence the root of $\chi_p(R_i)$ has expansion $c^p\,t_\lambda^v\,(1+\tau_2(d)\,t_\lambda+\mathcal{O}(t_\lambda^2))$ at infinity (see Theorem 2). The proposition now follows from the fact that R and $R_1, \ldots, R_o$ can be reduced mod p (see Proposition 9) and Lemma 12**(i)**.□

Some other possible improvements can be deduced from Theorem 2 : these ones are not used in the Maple implementation :
(I). Suppose we are searching for the e-solutions of $L \in C(x)[\tau]$ that have local type $(c, v, d+\mathbb{Z})$ at infinity. Let p be a good prime for L. Distinct singularities do not necessarily stay distinct after reduction mod p. However one can use the following approach to discard some combinations in Step **2** of Algorithm Search. Using the notation of Search : let $\mathcal{S} \subset \mathcal{R}(\chi_p(L))$ be the set of eigenvalues of the p-curvature matrix having

expansion $c^p\, t_\lambda^v\, (1 + \tau_2(d)\, t_\lambda + \mathcal{O}(t_\lambda^2))$ at infinity. For each $s \in \mathcal{S}$, one can compute the combinations in Step **2** of Algorithm Search that "match this root" as follows : factor s over $C'[p](\lambda)$ (let $s = V_1^{e_1} \cdots V_l^{e_l}$). For all $k \in \{1, \ldots, l\}$, consider all the couples (i, j) such that V_k divides $\tau_1(P_{i,j})$ and remove all the combinations such that the sum taken over all such (i, j) of the $m_{i,j}$ is not equal to e_k. After that, keep only the combinations matching at least one element of $\mathcal{S}$.

Remark 10. *This method to discard some combinations is very close to the one used in [CH04, Section 5.2] in the differential case.*

(II). From the proof of Theorem 2, if for a good prime p for L, we have an element $(c, v, d + \mathbb{Z}) \in \overline{g}_\infty(L)$ such that c or d can not be reduced mod p, or c reduces to 0 mod p, then this $(c, v, d + \mathbb{Z})$ can be thrown away.

Example 10. *Taking $p = 29$ and $p = 5$ which are good primes in Example 7 one sees that all elements of $\overline{g}_\infty(L)$ given in Example 8 can be thrown away, so the example has no hypergeometric solutions.*

3.11 Computation timings

A Maple implementation of HypSols by the second author has been available since 2001 on
`http://www.math.fsu.edu/~hoeij/maple.html` and is now included in Maple 9. To use it, do `with(LREtools):` and then call `hypergeomsols`. Older versions of Maple have an implementation of Petkovšek's algorithm (called here PetkoSols). This algorithm can still be called in Maple 9 by setting `_Env_old_hypergeomsols:=true` before calling `hypergeomsols`. A comparison on 36 examples can be found on
`http://www.math.fsu.edu/~hoeij/comparison_hypergeomsols`.
The tests show that HypSols is much faster than PetkoSols except on very small examples. We give one additional example :

$$L(u) \;=\; u(x+2) - (x+1)\,(2\,x^2 + 3\,x + 2)\,u(x+1) + (x^6 + 2\,x^5 + x^4 - 2)\,u(x) \;=\; 0.$$

Recall that $\mathrm{hyp}(r)$ is short notation for a solution u of the operator $\tau - r$. The operator L has a basis of two hypergeometric solutions, which are conjugated over $\mathbb{Q}$. If HypSols is given the field $C = \mathbb{Q}$ then the output of HypSols will contain only one solution $u(x) = \mathrm{hyp}(x^3 + x^2 + \sqrt{2})$ because the output is a basis *up to conjugation* over C. Note that HypSols needed to enter the "hard case" (Algorithm HSol) to find this solution. Since the user will expect `hypergeomsols` to return a basis of solutions, not a basis up to conjugation, Maple will compute all conjugates of $x^3 + x^2 + \sqrt{2}$ over $C = \mathbb{Q}$. The expression $\mathrm{hyp}(r)$ can be rewritten as $\prod_{k=0}^{x-1} r(k)$ but it can also be written as a product of Γ functions (Maple's choice depends on the factorization of r). The output of

```
with(LREtools):
L := u(x)*(x^6+2*x^5+x^4-2)-(x+1)*(2*x^2+3*x+2)*u(x+1)+u(x+2);
hypergeomsols(L, u(x), {}, output=basis);
```

in Maple 9 on this example is

$$\prod_{k=0}^{x-1}\left(k^3+k^2+\sqrt{2}\right), \quad \prod_{k=0}^{x-1}\left(k^3+k^2-\sqrt{2}\right).$$

The computation takes about 1.2 seconds (Pentium 733 MHz). In PetkoSols the example takes about 436 seconds. Mathematica's implementation of Petkovšek's algorithm finds no solutions because it is limited to the case where a_0 and a_n have no irreducible factors of degree > 2, see [PWZ96, Section 8.5].

Another comparison between the algorithm in [Ho99] and Petkovšek's algorithm was done in [We01, Section 6.3]. Here, the author points out the problem with the number of cases Petkovšek's algorithm must consider and the high cost of computing the sets $\overline{g}_p(L)$, most likely caused by not using truncated power series as in Section 4. The author also remarks that, unlike [Ho99], Petkovšek's algorithm may return too many (*i.e.*, linearly dependent) solutions.

3.12 An analogue algorithm for q-difference equations

In this section, we explain briefly how to develop the q-analogue of HypSols, for computing q-hypergeometric solutions of q-difference operators when q is not a root of unity and not zero. The q-analogue of Petkovšek's algorithm was given in [APP98].

We consider a field C of characteristic zero and its algebraic closure $\overline{C}$. To simplify the notations, we set $K := C(x)$.

Definition 47. *Let $q \in \overline{C}$, not zero and not a root of unity. A* q-difference operator *$L := \sum_{i=0}^{n} a_i\,\tau^i$ with $a_i = a_i(x) \in \overline{C}(x)$ is an operator that acts on $u = u(x)$ as follows :*

$$L(u(x)) = \sum_{i=0}^{n} a_i(x)\,u(q^i x).$$

In [PS97, Section 12.1], the authors prove the existence of a universal extension $\mathcal{V}_q$ for q-difference equations with coefficients in $\overline{C}(x)$.

Definition 48. *An expression $u(x) \in \mathcal{V}_q$ is called a* q-hypergeometric term *if it satisfies $u(qx)/u(x) \in \overline{C}(x)$.*

We are going to show that we can define a notion of local type at each point for first order q-difference operators (or q-hypergeometric terms) and construct sets containing the possible local types for possible first order right hand factors of q-difference operators. We shall also exhibit Fuchs' relations and show that the type of a first order q-difference operator (see Definition 49) only depends on its local types at each point.

3.12.1 Local types

While a difference operator has only one special singularity, namely at ∞, a q-difference operator has two special singularities : 0 and ∞. Consider the two following groups with multiplication $\circledS$ where $\circledS$ is defined as in Definition 30 :

$$K^*_{\circledS} := \{\tau - r \,|\, r \in K^*\} \quad \text{and} \quad K^*_R := \{\tau - \frac{\tau(r)}{r} \,|\, r \in K^*\}.$$

Note that $K^* \cong K^*_{\circledS}$ (resp. $K^*/C^* \cong K^*_R$) under $r \mapsto \tau - r$ (resp. $r \mapsto \tau - \tau(r)/r$).

Definition 49. *Let r in K^*. The* type of $\tau - r$ *is defined as the image of $\tau - r$ in $K^*_{\circledS}/K^*_R$.*

Special singularities

Here, we will define the local type at 0 and ∞ of $L = \tau - r$ with $r \in K^*$. Let $\langle q \rangle$ denote the multiplicative group generated by q (remember that q is not a root of unity and not zero).

Definition 50. *For $e \in \overline{C}^*$, denote $\overline{e} = \{e\, q^i \,|\, i \in \mathbb{Z}\}$, the image of e in $\overline{C}^*/\langle q \rangle$.*

Let

$$\mathcal{G}_0 = \mathcal{G}_\infty := \overline{C}^*/\langle q \rangle \;\times\; \mathbb{Z}.$$

We will use the additive notation for this group : $(\overline{c_1}, v_1) + (\overline{c_2}, v_2) = (\overline{c_1 c_2}, v_1 + v_2)$. The element $r \in K^*$ can both be written in the form $r = c_0\, x^{v_0}\,(1 + \mathcal{O}(x^1))$ and $r = c_\infty\, t^{v_\infty}\,(1 + \mathcal{O}(t^1))$ where $t := 1/x$, c_0, $c_\infty \in \overline{C}^*$ and v_0, $v_\infty \in \mathbb{Z}$.

Definition 51. *With these notations, the* local types *of $L = \tau - r$ at 0 and ∞ are respectively defined as :*

$$g_0(L) = (\overline{c_0}, v_0) \in \mathcal{G}_0,$$

and

$$g_\infty(L) = (\overline{c_\infty}, v_\infty) \in \mathcal{G}_\infty.$$

Note that g_0 (resp. g_∞) define homomorphisms from $K^*_{\circledS}/K^*_R$ to $\mathcal{G}_0$ (resp. $\mathcal{G}_\infty$). So $g_0(\tau - r)$ and $g_\infty(\tau - r)$ only depend on the type of $\tau - r$.

Other Singularities

Definition 52. *Let C be a field of characteristic zero and $L \in C[x][\tau]$ be a q-difference operator. Then $\alpha \in \overline{C}^*$ is a* problem point of L *if α is a root of $a_0(x)\, a_n(x/q^n)$.*
$\overline{\alpha} \in \overline{C}^*/\langle q \rangle$ is a singularity of L *if L has a problem point in $\overline{\alpha}$.*

Definition 53. *For $e \in \overline{C}^*$ and $r \in K^*$, we define the* valuation $v_e(r)$ of r at e *as the largest integer m such that $r/(x-e)^m \in \overline{C}[[x-e]]$. The valuation of 0 is ∞.*

Definition 54. *For $\overline{\alpha} \in \overline{C}^*/\langle q \rangle$, we define the group homomorphism $v_{\overline{\alpha}} : K^* \to \mathbb{Z}$ given by $v_{\overline{\alpha}}(r) = \sum_{e \in \overline{\alpha}} v_e(r)$. The* local type of $L = \tau - r$ at $\overline{\alpha}$ *is defined as*

$$g_{\overline{\alpha}}(L) = v_{\overline{\alpha}}(r) \in \mathbb{Z}.$$

It is easy to show that $g_{\overline{\alpha}}(\tau - r)$ only depends on the type of $\tau - r$.

3.12.2 Fuchs' relations

Let $\mathcal{G}_+$ be the additive group of functions $C^*/\langle q\rangle \to \mathbb{Z}$ with finite support and let

$$\mathcal{H} := \mathcal{G}_+ \times \mathcal{G}_0 \times \mathcal{G}_\infty.$$

Definition 55. *The* collection of local types *of $L = \tau - r$ is defined as*

$$g(L) := (\overline{\alpha} \mapsto g_{\overline{\alpha}}(L), g_0(L), g_\infty(L)) \in \mathcal{H}.$$

Lemma 14 (Fuchs' Relations). *Let $L = \tau - r \in K^*_{\circledS}$.*
If $g(L) := (\overline{\alpha} \mapsto g_{\overline{\alpha}}(L), (\overline{c_0}, v_0), (\overline{c_\infty}, v_\infty)) \in \mathcal{H}$ is the collection of local types of L, then we have

$$v_0 \;+\; v_\infty \;+ \sum_{\overline{\alpha}\in\overline{C}^*/\langle q\rangle} g(\overline{\alpha}) = 0 \quad \text{and} \quad \overline{c_0} \;=\; \overline{c_\infty} \prod_{\overline{\alpha}\in\overline{C}^*/\langle q\rangle} (\overline{-\alpha})^{g(\overline{\alpha})}. \tag{3.14}$$

Proof. One can check this for generators of the group $K^*_{\circledS}$, *i.e.*, the $\tau - c$ with $c \in \overline{C}^*$ and the $\tau - (x - e)$ with $e \in \overline{C}$.□

Let $\mathcal{H}_F$ be the subset of $\mathcal{H}$ containing all the elements satisfying Fuchs' relations given by Equation (3.14).

Proposition 11. *g induces a group isomorphism from $K^*_{\circledS}/K^*_R$ to $\mathcal{H}_F$.*

Proof. Let $g : K^*_{\circledS} \to \mathcal{H}$ be as in Definition 55. From the previous lemma, the image of g is contained in $\mathcal{H}_F$. The map is surjective since if $F = (f, (\overline{c_0}, v_0), (\overline{c_\infty}, v_\infty)) \in \mathcal{H}_F$, then $F = g(L)$ where $L = \tau - c_\infty\, x^{v_0}\,(x-\alpha_1)^{f(\alpha_1)}\cdots(x-\alpha_m)^{f(\alpha_m)}$. Here, c_∞ is a representant in $\overline{C}^*$ of $\overline{c_\infty}$ and $\alpha_1, \ldots, \alpha_m$ are representants in $\overline{C}^*$ for the $\overline{\alpha_i} \in \overline{C}^*/\langle q\rangle$ for which $f(\overline{\alpha_i}) \neq 0$, and $\overline{\alpha_i} \neq \overline{\alpha_j}$ if $i \neq j$. Furthermore, it is easy to check that such an L is in K^*_R only if $F = (0, (\overline{1}, 0), (\overline{1}, 0))$ which is the identity in $\mathcal{H}_F$. So the kernel of g is K^*_R.□

Consequently, the type of a q-difference operator of order one is determined by its local types. Now, as in the difference case, given $L \in K[\tau]$, one can effectively construct sets containing all possible local types of possible order one right hand factors of L. Then q-HypSols, the q-analogue of HypSols, follows naturally.

Chapitre 4

Solutions polynomiales d'équations différentielles linéaires

Introduction

En calcul formel, calculer les solutions polynomiales d'une équation différentielle linéaire est une étape centrale dans beaucoup d'algorithmes ; par exemple, les algorithmes classiques calculant les solutions rationnelles (voir [Br92, Ba99]) ou exponentielles (voir Chapitre 2) d'équations différentielles linéaires se ramènent au calcul de solutions polynomiales. De nos jours plusieurs algorithmes existent (voir [ABP95], [Ba99] - ou sa version scalaire dans [Ba97] et les références données dans ces papiers) et la variété des situations semble actuellement empêcher un algorithme donné d'être universellement meilleur que les autres. Le but dans ce chapitre est d'évaluer la pertinence des informations modulaires que l'on peut obtenir pour ce problème du calcul des solutions polynomiales d'équations différentielles linéaires.

Pour ce faire, nous construisons un algorithme modulaire pour calculer les solutions polynomiales d'équations différentielles linéaires. Cet algorithme procède comme suit :

- on calcule les solutions polynomiales modulo des nombres premiers p avec la version modulaire de l'un des algorithmes classiques ([ABP95] ou [Ba99]),
- on reconstruit les solutions polynomiales en caractéristique zéro en utilisant une combinaison de l'algorithme des restes chinois (voir [GG99, 5.4]) et de la reconstruction rationnelle (voir [GG99, 5.10] ou encore [CE95, Mo04]).

Contrairement aux algorithmes présentés dans les chapitres 2 et 3 qui utilisent l'information modulaire comme filtre, cet algorithme est complètement modulaire et, à notre connaissance, c'est le premier algorithme complètement modulaire calculant des solutions d'équations différentielles linéaires.

Notre algorithme modulaire ne peut pas être universellement meilleur que les algorithmes existants : en effet, nous savons que les algorithmes modulaires sont surtout efficaces pour résoudre des problèmes dans lesquels la taille des objets considérés dans

les calculs intermédiaires est significativement plus grande que celle de la sortie ce qui n'est, en général, pas le cas pour le calcul des solutions polynomiales d'équations différentielles linéaires. Pour confirmer cela et avoir une idée des cas dans lesquels on peut espérer que notre algorithme soit meilleur que ceux qui existent :

- nous étudions la complexité des deux algorithmes les plus utilisés à savoir celui d'Abramov, Bronstein et Petkovšek (voir [ABP95]) et celui de Barkatou (voir [Ba99]),
- nous analysons la complexité de notre algorithme modulaire et effectuons la comparaison. Ceci nous permet d'exhiber des classes naturelles d'opérateurs pour lesquels notre algorithme est plus efficace.

Notre analyse modulaire fournit au passage des résultats intéressants qui mènent à des tests de non-existence de solutions polynomiales non-nulles très utiles en pratique. Nous pouvons même obtenir mieux que des tests de non-existence puisque notre approche donne des heuristiques pour le calcul de la dimension de l'espace des solutions rationnelles ce qui peut s'avérer très utile pour certains problèmes en théorie de Galois différentielle. Notons encore que les avancées récentes de [Bos03] permettent d'améliorer encore ces résultats.

4.1 Préliminaires sur les solutions polynomiales

Dans un soucis de complétude, nous rappelons ici brièvement quelques faits bien connus concernant les solutions polynomiales d'équations différentielles linéaires. Pour plus de détails, nous renvoyons à [In26] ou [ABP95].

4.1.1 Équation aux récurrences associée

Soit $L = a_n\,\partial^n + \cdots + a_0$ un opérateur différentiel linéaire avec les a_i dans $\mathbb{Z}[x]$. Supposons que $a_0, \ldots, a_n$ sont premiers entre eux (dans $\mathbb{Q}[x]$) ce qui n'affecte pas la généralité de ce qui suit. Pour un tel L, définissons

$$A := -\min_i(\mathrm{val}_x(a_i) - i),\ B := \max_i(\deg_x(a_i) - i), \tag{4.1}$$

où, pour un polynôme f appartenant à $\mathbb{Z}[x]$, $\mathrm{val}_x(f)$ et $\deg_x(f)$ représentent respectivement la valuation x-adique et le degré en x de f.

On note alors

$$a_i = \sum_{j=-A}^{B} a_{i,j}\, x^{i+j},$$

avec les $a_{i,j}$ dans $\mathbb{Z}$ et pour $j \in \{-A, \ldots, B\}$, on pose

$$\alpha_j(k) := \sum_{i=0}^{n} a_{i,j}\, \frac{k!}{(k-i)!} \in \mathbb{Z}. \tag{4.2}$$

Si $y = \sum_{i=v}^{N} y_i\, x^i$ avec $y_v \neq 0$ et $y_N \neq 0$ est une solution polynomiale de L, alors

$$\begin{aligned} L(y) &= \sum_{i=v}^{N} y_i \left(\alpha_{-A}(i)\, x^{i-A} + \cdots + \alpha_B(i)\, x^{i+B}\right), \\ &= \sum_{i=v-A}^{N+B} \left(\alpha_{-A}(i+A)\, y_{i+A} + \cdots + \alpha_B(i-B)\, y_{i-B}\right)\, x^i, \end{aligned}$$

avec la convention $y_i = 0$ dès que $i < v$ ou $i > N$. Donc la relation $L(y) = 0$ implique que les y_i satisfont la récurrence linéaire

$$\alpha_{-A}(i+A)\, y_{i+A} + \cdots + \alpha_B(i-B)\, y_{i-B} = 0. \tag{4.3}$$

Pour $i = N + B$, l'équation (4.3) donne $\alpha_B(N)\, y_N = 0$. De même, pour $i = v - A$, nous obtenons $\alpha_{-A}(v)\, y_v = 0$. Par conséquent, puisque $y_N \neq 0$ (respectivement $y_v \neq 0$), le degré N (resp. la valuation v) de la solution polynomiale y est nécessairement une racine entière positive du polynôme α_B (resp. α_{-A}).

Définition 25. *Soit A et B définis par (4.1). On dit que α_B (resp. α_{-A}) est* le polynôme indiciel *de L en $x = \infty$ (resp. $x = 0$).*

Notons que les racines de α_B ne sont pas exactement ce que l'on appelle *exposants de L à l'infini* dans la littérature mais leurs opposés.

Exemple 15. *Considérons l'opérateur*

$$L_m = m\, x\, \partial^2 - x\, \partial + m^2,$$

où m est un entier naturel non-nul fixé. Pour cet opérateur, nous avons $A = -\min(1-2, 1-1, 0-0) = 1$ et $B = \max(1-2, 1-1, 0-0) = 0$. Écrivons l'équation aux récurrences linéaire associée à L : on a

$$\begin{aligned} L(\textstyle\sum_i y_i\, x^i) &= m\, x \textstyle\sum_i i\,(i-1)\, y_i\, x^{i-2} - x \textstyle\sum_i i\, y_i\, x^{i-1} + m^2 \textstyle\sum_i y_i\, x^i, \\ &= \textstyle\sum_i [m\, i\,(i+1)\, y_{i+1} + (m^2 - i)\, y_i] x^i. \end{aligned}$$

La récurrence linéaire associée à L s'écrit donc :

$$m\, i\,(i+1)\, y_{i+1} + (m^2 - i)\, y_i = 0.$$

On voit ainsi que le polynôme indiciel de L en $x = \infty$ est $\alpha_0(T) = m^2 - T$ et donc une éventuelle solution polynomiale de L ne peut avoir que m^2 pour degré. Le polynôme indiciel en $x = 0$ est $\alpha_{-1}(T) = m\, T\,(T-1)$ et la valuation d'une éventuelle solution polynomiale de L est nécessairement 0 ou 1. Notons qu'ici on peut voir que cette valuation ne peut être que 1 : en effet, pour $i = 0$, l'équation aux récurrences linéaire donne $m^2\, y_0 = 0$ et donc le coefficient constant y_0 d'une solution polynomiale de L_m est nécessairement nul.

4.1.2 Degré des solutions polynomiales

Notons $\mathcal{I}(L,\infty)$ l'ensemble (fini) contenant toutes les racines entières positives (distinctes) du polynôme indiciel de L à l'infini. D'après la sous-section précédente, le degré d'une solution polynomiale de L doit appartenir à cet ensemble $\mathcal{I}(L,\infty)$. De la même façon, la valuation d'une solution polynomiale de L doit être un élément de l'ensemble des racines entières positives (distinctes) du polynôme indiciel de L en zéro. Par la suite, nous supposerons que la borne inférieure pour la valuation d'une éventuelle solution polynomiale, *i. e.*, la plus petite racine entière positive de α_{-A}, est 0. Ceci n'affecte pas la généralité de ce qui suit puisque cela est toujours satisfait à un changement de variable près.

Alors que l'algorithme de [ABP95] utilise seulement une borne sur le degré d'éventuelles solutions polynomiales, notre algorithme (comme celui de [Ba97]) nécessite la connaissance de l'ensemble fini $\mathcal{I}(L,\infty)$ tout entier. Une borne sur le degré des solutions polynomiales est évidemment donnée par le plus grand élément de $\mathcal{I}(L,\infty)$.

À partir de $\mathcal{I}(L,\infty)$, nous pouvons aussi déduire une borne sur la dimension de l'espace des solutions polynomiales non-nulles de L : son cardinal $\#\mathcal{I}(L,\infty)$ (qui est $\leq n$). Ceci donne en particulier un premier test pour détecter la non-existence de solutions polynomiales non-nulles. Bien que très simple, ce test peut être utile en pratique : par exemple, pour un opérateur différentiel linéaire L choisi au hasard, $\mathcal{I}(L,\infty)$ est vide ce qui suffit pour prouver que L n'a pas de solutions polynomiales non-nulles. Notons encore que ce test peut être amélioré en considérant les polynômes indiciels en toutes les singularités de L : si le polynôme indiciel de L en une singularité quelconque de L n'a pas de racines entières positives, alors L n'a pas de solutions polynomiales non-nulles (ceci est remarqué dans [Ba97, Remark 1]).

4.2 Calculer les solutions polynomiales en caractéristique zéro

Dans cette partie divisée en trois sous-sections, nous rappelons 3 méthodes pour calculer les solutions polynomiales d'une équation différentielle linéaire.

4.2.1 Résoudre le système linéaire associé

Une première méthode (naïve) pour calculer les solutions polynomiales d'un opérateur différentiel $L \in \mathbb{Z}[x][\partial]$ est la suivante : soit N une borne sur le degré d'éventuelles solutions polynomiales de L. On cherche donc des solutions polynomiales de la forme $y = \sum_{i=0}^{N} y_i\, x^i$ avec les y_i à déterminer dans $\mathbb{Q}$. L'équation $L(y) = 0$ donne alors un système linéaire de dimension $(N + A + B + 1) \times (N + 1)$ pour les y_i : en effet, il nous suffit de faire varier i de $-A$ à $N + B$ dans (4.3) pour obtenir toutes les équations faisant intervenir $y_0, \ldots, y_N$. Calculer les solutions polynomiales revient à résoudre ce système. Si l'on regarde d'un peu plus près, alors on s'aperçoit que ce système est de la forme suivante :

$$\begin{pmatrix} \times & 0 & 0 & \cdots & 0 \\ \times & \times & 0 & \cdots & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \times & \cdots & \cdots & \times & 0 \\ \times & \cdots & \cdots & \cdots & \times \\ \vdots & & & & \vdots \\ \times & \cdots & \cdots & \cdots & \times \\ 0 & \times & \cdots & \times & \times \\ \vdots & 0 & \ddots & & \vdots \\ \vdots & & \ddots & \times & \times \\ 0 & 0 & \cdots & 0 & \times \end{pmatrix}.$$

Les « $\times$ » représentent ici certains des $\alpha_i(k)$ définis dans la sous-section 4.1.1 (voir Équation 4.2).

La forme particulière de ce système rend sa résolution plus simple que celle d'un système quelconque et donc la complexité de la résolution d'un tel système ne sera pas en $\mathcal{O}(N^3)$ opérations dans $\mathbb{Q}$. En effet, pour résoudre un système sous cette forme, aucune manipulation sur la matrice (type Gauss) n'est nécessaire et il suffit juste de résoudre les équations fournies par chaque ligne les unes à la suite des autres : le fait d'utiliser toute la structure de ce système pour le résoudre constitue l'idée principale de l'algorithme d'Abramov, Bronstein et Petkovšek décrit dans la section suivante.

Exemple 16. *Reprenons l'opérateur de l'exemple 15 et écrivons le système que l'on obtient dans ce cas. Pour simplifier, supposons* $m = 2$ *: on a alors* $N = 4$ *comme borne sur le degré d'éventuelles solutions polynomiales. D'après ce qui précède, nous savons que le système est de taille* $(N + A + B + 1) \times (N + 1)$ *donc ici* 6×5*. Pour l'écrire il nous suffit de faire varier* i *de* -1 *à* 4 *dans la récurrence associée (voir Exemple 15) : on obtient le système linéaire suivant :*

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 \\ 0 & 3 & 4 & 0 & 0 \\ 0 & 0 & 2 & 12 & 0 \\ 0 & 0 & 0 & 1 & 24 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

En le résolvant, on trouve $[y_0, y_1, y_2, y_3, y_4] = [0, -192, 144, -24, 1]$ de sorte qu'une base des solutions polynomiales de L_2 est $\{-192\,x + 144\,x^2 - 24\,x^3 + x^4\}$.

4.2.2 L'algorithme d'Abramov, Bronstein et Petkovšek

Dans cette partie, nous rappelons brièvement l'algorithme, noté ici ABP, donné dans [ABP95] par Abramov, Bronstein et Petkovšek. Cet algorithme est basé sur la résolution

de l'équation aux récurrences (4.3). Nous étudions ensuite sa complexité.

Pour la description de l'algorithme, nous nous plaçons dans la situation de [ABP95] c'est-à-dire que nous supposons que 0 est un point ordinaire de l'opérateur différentiel L : les racines du polynôme indiciel α_{-A} de L en 0 sont alors $0, \ldots, n-1$. Si 0 n'est pas ordinaire, alors il nous faut faire un changement de variable.

Remarque 14. *Il est possible de modifier cet algorithme pour le rendre correct même si on ne se place pas en un point ordinaire. Pour ceci il nous faut prendre en compte les contraintes linéaires sur les conditions initiales qui nous sont imposées par l'équation aux récurrences. Notons qu'en pratique il est souvent plus avantageux de se placer en un point non-ordinaire puisque l'équation indicielle possède alors (en général) un nombre de racines entières positives strictement inférieur à n.*

Algorithme ABP

Entrée : $L = \sum_{i=0}^{n} a_i\, \partial^i \in \mathbb{Q}[x][\partial]$ satisfaisant $a_n(0) \neq 0$ et une borne N sur le degré d'éventuelles solutions polynomiales de L.
Sortie : une base des solutions polynomiales de L de degré inférieur ou égal à N.

1. Pour $j \in \{-A, \ldots, B\}$, calculer les α_j,
2. Pour $k \in \{0, \ldots, N + A + B\}$, calculer les $\alpha_j(k)$ (évaluation multi-points [1]),
3. Pour i de 1 à n faire :
   3a. Soit $y_i = \sum_{j=0}^{N+A+B} y_{i,j}\, x^j$ où pour $j \in \{0, \ldots, n-1\}$, $y_{i,j} = \delta_{i-1}^{j}$.

   3b. Utiliser la récurrence (4.3) pour calculer les $y_{i,j}$ pour $j \in \{0, \ldots, N+A+B\}$,

4. Résoudre (en les c_i) le système linéaire donné par les $\sum_{i=1}^{n} c_i\, y_{i,N+j} = 0$ pour $j \in \{1, \ldots, A + B\}$,
5. Retourner les $\sum_{i=1}^{n} c_i\, y_i$ pour tout $(c_1, \ldots, c_n)$ trouvé à l'étape 4.

Exemple 17. *Regardons l'évolution de cet algorithme sur l'opérateur L_2 de l'exemple 15. Le point 0 n'est pas ordinaire donc nous devons effectuer un changement de variable : envoyons par exemple x sur $x - 1$. La récurrence devient alors*

$$-2\,(i+2)\,(i+1)\,y_{i+2} + (i+1)\,(2\,i+1)\,y_{i+1} + (4-i)\,y_i = 0.$$

Nous avons maintenant $A = 2$, $B = 0$ et $N = 4$. Posons $y_1 = \sum_{j=0}^{6} y_{1,j}\, x^j$ avec $y_{1,0} = 1$, $y_{1,1} = 0$ et $y_2 = \sum_{j=0}^{6} y_{2,j}\, x^j$ avec $y_{2,0} = 0$, $y_{2,1} = 1$. La récurrence nous donne alors

[1] Dans [Bos03, Chapter 5] (voir aussi [BLS03]), l'auteur montre qu'un polynôme P de degré d à coefficients dans un corps $\mathcal{F}$ peut être évalué en $d+1$ points de $\mathcal{F}$ en $\mathcal{O}(M(d)\,\log(d))$ opérations dans $\mathcal{F}$ où $M(d)$ est le coût de la multiplication de deux polynômes univariés de degré d ; la contribution est l'amélioration de la constante masquée par la notation $\mathcal{O}$. Ce résultat de complexité sur l'évaluation multi-points sera utilisé plusieurs fois par la suite.

d'un côté $y_{1,2} = 1$, $y_{1,3} = 1/2$, $y_{1,4} = 19/48$, $y_{1,5} = 139/480$, $y_{1,6} = 139/640$ *et de l'autre* $y_{2,2} = 1/4$, $y_{2,3} = 3/8$, $y_{2,4} = 49/192$, $y_{2,5} = 361/1920$, $y_{2,6} = 361/2560$. *Nous devons ensuite résoudre le système* $M\,{}^t(c_1, c_2) = {}^t(0,0)$ *avec* $M = \begin{pmatrix} \frac{139}{480} & \frac{361}{1920} \\ \frac{139}{640} & \frac{361}{2560} \end{pmatrix}$. *On trouve* $c_1 = c$ *et* $c_2 = -556/361\,c$ *et une base de solutions polynomiales de l'opérateur en* $x-1$ *est donnée par* $\{y_1 - 556/361\,y_2\}$. *Nous effectuons ensuite le changement de variable inverse* ($x \mapsto x+1$) *et nous retrouvons la base de solutions polynomiales de* L_2 *trouvée dans l'exemple 16 à savoir* $\{x^4 - 24\,x^3 + 144\,x^2 - 192\,x\}$.

Proposition 14. *Soit* $L \in \mathbb{Q}[x][\partial]$ *un opérateur différentiel d'ordre* n, A *et* B *définis par (4.1) et* N *une borne sur le degré d'éventuelles solutions polynomiales de* L. *L'algorithme* ABP *donné ci-dessus est correct et calcule une base des solutions polynomiales de* L *en* $\mathcal{O}(N\,n^2\,(A+B))$ *opérations dans* $\mathbb{Q}$.

Démonstration. Pour la correction de cet algorithme, nous renvoyons à [ABP95].
Dans le cas où 0 n'est pas un point ordinaire, nous devons faire un changement de variable $x \mapsto x + a$ avec $a \in \mathbb{Z}$. D'après la formule $a_i = \sum_{i=-A}^{B} a_{i,j}\,x^{i+j}$ ceci peut se faire en $\mathcal{O}((n+1)\,(A+B)\,(n+B))$ opérations dans $\mathbb{Q}$. Pour la suite de l'analyse de complexité, nous basons notre preuve sur [Bos03, 10.2]. Dans l'étape 1, calculer les polynômes α_i revient à écrire L en fonction de la dérivation d'Euler $\delta := x\,\frac{d}{dx}$. En effet, si $L \in \mathbb{Q}[x][\delta]$, alors la relation $\delta^j\,x^k = k^j\,x^k$ donne directement les α_i. Écrire L en fonction de δ peut se faire en $\mathcal{O}(\mathcal{M}(n)\,(A+B)\,\log(n))$ opérations dans $\mathbb{Q}$ où $\mathcal{M}(d)$ est le coût de la multiplication de deux polynômes univariés de degré $\leq d$ (note : avec la FFT, ceci peut se faire en $\mathcal{O}(d\,\log(d)\,\log(\log(d)))$ opérations de corps ; voir [GG99, Theorem 8.23]). Nous savons (voir [Bos03] ou [BLS03]) qu'avec les techniques de l'évaluation multi-points, un polynôme P de degré d à coefficients dans un corps $\mathcal{F}$ peut être évalué en $d+1$ points de $\mathcal{F}$ en $\mathcal{O}(M(d)\,\log(d))$ opérations dans $\mathcal{F}$ où $M(d)$ est le coût de la multiplication de deux polynômes de degré d. Par conséquent, dans l'étape 2, nous pouvons évaluer les $A+B+1$ polynômes α_j de degré $\leq n$ en les points $0, \ldots, N+A+B$ en $\mathcal{O}((A+B+1)\,\frac{N+A+B}{n}\,\mathcal{M}(n)\,\log(n))$ opérations dans $\mathbb{Q}$. Dans l'étape 3b, nous calculons les $N+A+B-n$ coefficients manquants de y_i en $\mathcal{O}((N+A+B-n)\,(A+B))$ opérations dans $\mathbb{Q}$. À l'étape 4, nous avons à résoudre un système linéaire de dimensions $(A+B) \times n$: or comme nous sommes en un point régulier $A = n$ et nous avons donc plus d'équations que d'inconnus : la résolution du système peut donc se faire en $\mathcal{O}((A+B)\,n^2)$ opérations dans $\mathbb{Q}$. □

4.2.3 L'algorithme de Barkatou

Le troisième algorithme calculant les solutions polynomiales d'équations différentielles linéaires que nous rappelons ici est celui donné par Barkatou dans [Ba97] qui n'est autre que la version scalaire de celui proposé dans [Ba99]). Cet algorithme est ici présenté en détails et sa complexité est étudiée. Un algorithme utilisant le même style d'idée est présenté dans [AK91].

Notations et idée de l'algorithme

Commençons par fixer quelques notations. Soit

$$L = \sum_{i=0}^{n} a_i \, \delta^i,$$

avec $\delta = x \frac{d}{dx}$ et $a_i \in \mathbb{Q}(x)$. Dans ce qui suit, le degré d'une fraction rationnelle désigne la différence entre le degré de son numérateur et celui de son dénominateur. Notons

$$B = \max_i(\deg_x(a_i)),$$

(remarque : comme L est ici écrit en fonction de la dérivation d'Euler, ce B correspond bien au B défini dans (4.1)) et

$$a_i(x) = \bar{a}_i \, x^{\deg_x(a_i)} + \cdots,$$

avec $\bar{a}_i \in \mathbb{Q}$ ($\bar{a}_i = 0$ si $a_i = 0$) et où « $\cdots$ » désigne une somme de termes de degré strictement inférieur à $\deg_x(a_i)$. On a alors,

$$L(x^\lambda) = \alpha_B(\lambda) \, x^{\lambda+B} + \cdots + \alpha_{-A}(\lambda) \, x^{\lambda-A},$$

où A est défini par $-A = \min_i(\mathrm{val}_x(a_i))$ et, avec les notations précédentes,

$$\alpha_B(k) = \sum_{i=0}^{n} \bar{a}_i \, k^i$$

est le polynôme indiciel de L à l'infini.

Pour simplifier la description de l'algorithme, nous supposons que $B = 0$ ce qui n'enlève rien à la généralité : en effet si $B \neq 0$, alors il nous suffit de remplacer L par $x^{-B} \, L$ et b par $x^{-B} \, b$ dans tout ce qui suit.

On a alors le lemme suivant :

Lemme 16 ([Ba97], Lemma 1). *Avec les notations précédentes et en supposant $B = 0$, on a : pour tout $z \in \mathbb{Q}(x)$, $\deg_x(L(z)) \leq \deg_x(z)$ avec égalité si, et seulement si, $\alpha_B(\deg_x(z)) \neq 0$.*

On considère ici des équations non-homogènes $L(y) = b$ avec $b \in \mathbb{Q}(x)$. Même si dans ce chapitre c'est uniquement le cas homogène qui nous intéresse, la manière dont cet algorithme procède nous amène de toute façon à considérer des équations non-homogènes (l'algorithme ABP fonctionne lui aussi dans le cas non-homogène - voir [ABP95]). Soit $y \in \mathbb{Q}[x]$ et écrivons

$$y = c \, x^\mu + z,$$

avec $c \in \mathbb{Q}^*$ et $\mu \in \mathbb{N}$. On a alors

$$L(y) = b \Leftrightarrow L(z) = b - c\,L(x^\mu).$$

L'idée à la base de l'algorithme est alors la suivante : essayer de déterminer $c \in \mathbb{Q}^*$ et $\mu \in \mathbb{N}$ tels que $L(y) = b$ et $\deg_x(z) < \deg_x(y) = \mu$. Si on trouve un tel couple (c, μ) alors on recommence avec z à la place de y et l'équation $L(z) = b - c\,L(x^\mu)$ à la place de $L(y) = b$ et ainsi de suite jusqu'à ce que l'on obtienne une équation de la forme

$$L(w) = b - L(\sum_{\text{finie}} c_i\, x^{\mu_i})$$

qui n'a pas de solutions polynomiales (pas de choix possible pour c et μ vérifiant les conditions demandées). Par conséquent, l'équation de départ $L(y) = b$ aura une solution polynomiale non-nulle si, et seulement si, $L(\sum_{\text{finie}} c_i\, x^{\mu_i}) = b$.

Comment choisir $c \in \mathbb{Q}^*$ et $\mu \in \mathbb{N}$ tels que $L(c\,x^\mu + z) = b$ et $\deg_x(z) < \mu$?

Notons $b = b_0 + \sum_{i=1}^m c_i\, b_i$ avec $m \leq n$ et les $b_i \in \mathbb{Q}(x)$: au cours de l'algorithme le second membre de notre équation peut contenir des paramètres c_i. Soit

$$\mathcal{R} := \{\lambda \in \mathbb{N} \,|\, \alpha_B(\lambda) = 0\},$$

l'ensemble des racines entières positives du polynôme indiciel α_B de L à l'infini. Notons η le degré de b et

$$b = \bar{b}\,x^\eta + \cdots,$$

avec la convention $\bar{b} = 0$ si $\eta = -\infty$.

Si $y = c\,x^\mu + z$, alors

$$\begin{array}{rcl} L(y) = b & \Leftrightarrow & L(z) = b - c\,L(x^\mu), \\ & \Leftrightarrow & L(z) = \bar{b}\,x^\eta - c\,\alpha_B(\mu)\,x^\mu + \cdots. \end{array} \tag{4.4}$$

D'après le lemme 16, si un couple (c, μ) est solution de notre problème, alors $\deg_x(L(z)) \leq \deg_x(z) < \mu$. Pour simplifier les notations, posons

$$E := \bar{b}\,x^\eta - c\,\alpha_B(\mu)\,x^\mu + \cdots.$$

Trois cas sont à distinguer :

1) Si $[\mathcal{R} \neq \emptyset$ et $\eta < \max \mathcal{R}]$, alors il nous suffit de prendre $\mu = \max \mathcal{R}$ et c paramètre arbitraire. Par la suite c est considéré comme un paramètre. On retire alors μ de l'ensemble $\mathcal{R}$.

2) Si $[\mathcal{R} = \emptyset$ et $\eta < 0]$, alors, pour tout $\mu \geq 0$, le degré du second membre de E est égal à μ et il n'existe donc pas de couple (c, μ) solution de notre problème.

3) Si [$\mathcal{R} \neq \emptyset$ et $\eta \geq \max \mathcal{R}$] ou [$\mathcal{R} = \emptyset$ et $\eta \geq 0$], alors le seul choix possible pour μ est $\mu = \eta$ (car si $\mu > \eta$, alors le degré de E est égal à μ). Avec $\mu = \eta$, l'équation 4.4 donne

$$L(z) = (\bar{b} - c\,\alpha_B(\eta))\,x^\eta + \cdots,$$

et donc nécessairement

$$\bar{b} - c\,\alpha_B(\eta) = 0. \tag{4.5}$$

Cette dernière équation admet une solution $c \in \mathbb{Q}^*$ si, et seulement si, [$\alpha_B(\eta) \neq 0$] ou [$\alpha_B(\eta) = 0$ et $\bar{b} = 0$]. Ici encore, trois cas sont à distinguer :

3.1) Si $\alpha_B(\eta) \neq 0$, il suffit de prendre $c = \frac{\bar{b}}{\alpha_B(\eta)}$.

3.2) Si $\alpha_B(\eta) = 0$ et si le système linéaire en les c_i donné par $\bar{b}(c_1, \ldots, c_m) = 0$ a une solution, alors il suffit de prendre c paramètre arbitraire. Dans ce cas, on retire $\mu = \eta$ de l'ensemble $\mathcal{R}$.

3.3) Si $\alpha_B(\eta) = 0$ et si le système linéaire en les c_i donné par $\bar{b}(c_1, \ldots, c_m) = 0$ n'a pas de solution, alors il n'existe pas de couple (c, μ) solution de notre problème.

Notons que ce procédé se termine puisqu'à chaque étape, soit le degré du second membre décroit, soit le nombre d'éléments de $\mathcal{R}$ diminue.

Lorsque ce procédé se termine (c'est-à-dire que nous sommes dans le cas où il n'existe pas de c et μ répondant au problème), nous avons une équation de la forme $L(w) = b - L(\sum_{\text{finie}} c_i\,x^{\mu_i})$ (où les c_i sont des paramètres arbitraires) qui n'a pas de solutions polynomiales. Dans ce cas $\sum_{\text{finie}} c_i\,x^{\mu_i}$ est la solution polynomiale générale de l'équation de départ si, et seulement si, $L(w) = 0$. Ce $L(w)$ s'écrit $\sum \frac{f_i\,x^i}{g(x)}$ où les f_i sont des polynômes de degré 0 ou 1 en les paramètres c_i et il ne nous reste donc plus qu'à résoudre le système en les c_i formé par les équations $f_i = 0$.

Exemple 18. *Comme nous l'avons fait pour l'algorithme* ABP *dans la sous-section précédente, regardons comment évolue l'algorithme de Barkatou sur l'opérateur L_2 donné dans l'exemple 15. On a ici $B = 0$ de telle sorte qu'aucune modification préliminaire sur l'opérateur n'est nécessaire. D'après l'exemple 15, l'ensemble $\mathcal{R}$ défini ci-dessus est ici égal à $\{4\}$. Initialement nous avons donc $\max \mathcal{R} = 4$ et $\eta = -\infty$ de telle sorte que nous passons dans le cas 1 (voir ci-dessus). Nous posons donc $\mu_1 = 4$ et c_1 constante arbitraire. En posant $y = c_1\,x^4 + z$, nous sommes conduits à examiner l'équation $L(z) = -c_1\,L(x^4) = -24\,c_1\,x^3$. Le degré du second membre est donc égal à 3. Maintenant $\mathcal{R}$ est vide et $\eta = 3 \geq 0$ donc nous passons dans le cas 3. De plus, nécessairement, $\alpha_B(3) \neq 0$ donc nous passons dans 3.1. Il vient donc $z = -24\,c_1\,x^3 + w$ et w satisfait $L(w) = 288\,c_1\,x^2$. Ensuite nous repassons dans 3.1 et donc $w = 144\,c_1\,x^2 + v$ avec $L(v) = -576\,c_1\,x$. À l'étape suivante, toujours en passant par le cas 3.1, nous avons $v = -192\,c_1\,x + u$ où u vérifie $L(u) = 0$. Nous en arrivons donc au point où $\mathcal{R} = \emptyset$ et $\eta < 0$ de telle sorte que l'algorithme se termine puisque nous sommes dans le cas 2. Nous avons donc construit l'éventuelle solution polynomiale générale $y = c_1\,x^4 - 24\,c_1\,x^3 + 144\,c_1\,x^2 - 192\,c_1\,x$. À la fin nous avons $L(y) = 0$ et nous n'avons donc aucune condition à vérifier de telle sorte que $\{x^4 - 24\,x^3 + 144\,x^2 - 192\,x\}$ est une base de solutions polynomiales de L_2.*

Nous pouvons résumer toute l'explication donnée ci-dessus par l'algorithme suivant :

Algorithme Barkatou

Entrée : $L \in \mathbb{Q}(x)[\delta]$.
Sortie : la solution polynomiale générale de $L(y) = 0$.

1. Si $B \neq 0$, alors remplacer L par $x^{-B}\, L$.
2. Calculer $\alpha_B(= \alpha_0)$. Soit $\mathcal{R} := [u \in \mathbb{N} \,|\, \alpha_B(u) = 0]$ et $r := \#\mathcal{R}$.
3. Si $r = 0$, alors retourner 0. Sinon :
4. Poser sol $:= 0$, $\eta := -\infty$ et sm $:= 0$.
5. Tant que $\eta > 0$ ou $r > 0$, faire :
   - 5a. Si $r > 0$ et $\mathcal{R}[r] > \eta$, alors :
     sol $:=$ *sol* $+ c[r]\, x^{\mathcal{R}[r]}$,
     sm $:=$ sm $- c[r]\, L(x^{\mathcal{R}[r]})$,
     $\eta :=$ deg(sm) et $r := r - 1$.
   - 5b. Sinon, si $r > 0$ et $\mathcal{R}[r] = \eta$, alors :
     lcb $:=$ lcoeff(sm, x),
     Résoudre lcb $= 0$ en les c_i,
     S'il n'y a pas de solution, alors poser $\eta := -1$ et $r := 0$ (stop).
     Sinon :
     Assigner aux c_i les valeurs trouvées,
     sol $:=$ sol $+ c[r]\, x^{\mathcal{R}[r]}$,
     sm $:=$ sm $- c[r]\, L(x^{\mathcal{R}[r]})$,
     $\eta :=$ deg(sm) et $r := r - 1$.
   - 5c. Sinon :
     lcb $:=$ lcoeff(sm, x),
     $w :=$ lcb$/\alpha_0(\eta)$,
     sol $:=$ sol $+ w\, x^{\eta}$,
     $\eta :=$ deg(sm).
6. Si sm $\neq 0$, alors :
   - 6a. Résoudre le système en les c_i formé par les coefficients du numérateur de sm.
   - 6b. Si ce système n'a pas de solutions, alors retourner 0.
   - 6c. Sinon, assigner aux c_i les valeurs trouvées.
7. Retourner sol.

Proposition 15. *Soit $L \in \mathbb{Z}[x][\partial]$ un opérateur différentiel linéaire d'ordre n, A et B définis par (4.1) et N une borne sur le degré d'éventuelles solutions polynomiales de L. L'algorithme de Barkatou donné ci-dessus est correct et calcule une base de solutions polynomiales de L de degré inférieur ou égal à N en au plus $\mathcal{O}(N\, n^2\, (A+B))$ opérations dans $\mathbb{Q}$.*

Démonstration. Pour la correction de cet algorithme, nous renvoyons à [Ba97].
Le nombre d'opérations faites par cet algorithme récursif dépend fortement du nombre de passage dans la sous-procédure cherchant c et μ (nombre de passage dans la boucle « tant que » dans l'algorithme ci-dessus). Ce nombre de passage correspond au nombre de coefficients non-nuls dans l'écriture de la solution polynoniale générale de L. Ici nous allons nous intéresser au cas extrême où nous avons une base de solutions en séries dans $\mathbb{Q}[[1/x]][x]$ avec des parties polynomiales denses ($N+1$ coefficients non-nuls) mais aucune solution polynomiale non-nulle : ceci nous donnera donc une estimation de la complexité maximale de l'algorithme. Pour pouvoir comparer cet algorithme aux autres, nous devons supposer que l'opérateur de départ est écrit en la dérivation ∂ et que nous avons une équation homogène c'est-à-dire qu'initialement le second membre b est nul. Il nous faut donc tout d'abord écrire l'opérateur en fonction de la dérivation d'Euler ce qui se fait en $\mathcal{O}(\mathcal{M}(n)\,(A+B)\,\log(n))$ opérations dans $\mathbb{Q}$ où $\mathcal{M}(d)$ est le coût de la multiplication de deux polynômes de degré $\leq d$ (voir la preuve de la proposition 14). Notons que ceci est de toute façon nécessaire pour calculer les polynômes α_i définis dans la partie 4.1.1. Ensuite, au cours de l'algorithme nous aurons besoin d'évaluer les $L(x^\mu)$ ou $L(x^\eta)$ pour toute racine entière positive μ de α_B et pour les degrés η des seconds membres qui apparaissent au cours de l'algorithme : ces μ et η sont toujours $\leq N$. Pour ceci nous utilisons encore une fois les techniques de l'évaluation multi-points (voir [Bos03] ou [BLS03]) qui impliquent qu'un polynôme P de degré d à coefficients dans un corps $\mathcal{F}$ peut être évalué en $d+1$ points de $\mathcal{F}$ en $\mathcal{O}(M(d)\,\log(d))$ opérations dans $\mathcal{F}$ où $M(d)$ est le coût de la multiplication de deux polynômes de degré d. Par conséquent, nous pouvons évaluer au préalable les $A+B+1$ polynômes α_j de degré $\leq n$ en $0,\ldots,N$ en $\mathcal{O}((A+B+1)\,\frac{N}{n}\,\mathcal{M}(n)\,\log(n))$ opérations dans $\mathbb{Q}$. Maintenant, nous passons $N+1$ fois dans la sous-procédure cherchant c et μ et la solution générale sous forme de série dépend de n constantes arbitraires. Cela signifie que nous passons n fois dans l'étape 5a ou 5b et $N+1-n$ fois dans le cas 5c. Comme l'étape 5b est plus coûteuse que 5a, supposons que nous passons $n-1$ fois dans 5b et 1 fois dans 5a : lors du premier passage dans la boucle « tant que », nous passons forcément dans 5a puisque le degré du second membre est initialement $-\infty$ (notons que, vu les évaluations préliminaires, ce premier passage a un coût constant). Dans 5b, nous devons résoudre une équation linéaire en les c_i : résoudre cette équation signifie ici écrire le dernier paramètre arbitraire introduit en fonction des précédents de telle sorte que ceci coûte au plus $\mathcal{O}(n)$ opérations dans $\mathbb{Q}$ puisque qu'au maximum nous aurons n paramètres. La mise à jour du second membre a un coût constant puisque nous avons déjà évalué les α_i. Ensuite, nous devons passer $N+1-n$ fois dans 5c et toujours grâce au fait que les évaluations ont été faites au préalable ceci possède un coût constant. À la fin, il nous reste à résoudre le système formé par les $f_i = 0$ (où les f_i sont les coefficients du numérateur du second membre final). Ce système possède n inconnues (les c_i) et le nombre d'équations (le degré du numérateur du second membre final) est majoré par N : sa résolution coûte donc $\mathcal{O}(N\,n^2)$ opérations dans $\mathbb{Q}$. □

4.3 Calculer les solutions polynomiales en caractéristique p

Dans cette partie nous étudions le problème du calcul des solutions polynomiales non-nulles d'un opérateur différentiel en caractéristique p. Diverses stratégies sont envisagées : tout d'abord en remarquant qu'en caractéristique p, l'existence de solutions polynomiales est équivalente à l'existence de solutions rationnelles, nous proposons un premier algorithme. Ensuite nous montrons comment adapter les algorithmes de la section précédente à la caractéristique p.

4.3.1 À partir des solutions rationnelles

Soit $\mathbb{F}_p$ le corps fini à p éléments. Nous savons (voir Chapitre 1) que $\mathbb{F}_p(x)$ muni de la dérivation usuelle $' = \frac{d}{dx}$ est un corps différentiel ayant pour corps de constantes $\mathbb{F}_p(x^p)$. Une conséquence immédiate est qu'en caractéristique p, l'existence de solutions rationnelles est équivalente à l'existence de solutions polynomiales. Pour calculer, les solutions rationnelles d'une équation différentielle linéaire à coefficients dans $\mathbb{F}_p(x)$, nous disposons de l'algorithme suivant :

Algorithme RatSols-Mod$p$

Entrée : $L = \sum_{i=0}^{n} a_i\, x^i \in \mathbb{F}_p[x][\partial]$.
Sortie : une base de solutions rationnelles de L.

1. Calculer la matrice $p \times p$ de l'opérateur différentiel linéaire L dans la $\mathbb{F}_p(x^p)$-base $(1, x, \ldots, x^{p-1})$ de $\mathbb{F}_p(x)$ (voir le chapitre 1 pour des détails),
2. Calculer le noyau de cette matrice,
3. Pour chaque $(u_0, \ldots, u_{p-1})$ dans ce noyau, retourner $\sum_{i=0}^{p-1} u_i\, x^i$.

Lemme 17. *L'algorithme RatSols-Modp donné ci-dessus est correct et calcule une base de solutions rationnelles de L en $\mathcal{O}(n\,(A+B)\,\max(l,p)^2\,p^2)$ opérations dans $\mathbb{F}_p$ où l est une borne sur le degré (en x) des coefficients a_i de L.*

Remarquons que cette complexité est différente de celle donnée au chapitre 1 car nous prenons ici en compte le coût de la construction de la matrice.

Démonstration. La correction de l'algorithme découle directement des résultats de la section 2 du chapitre 1. Dans l'étape 1, le calcul de chaque ligne de la matrice nécessite le calcul de $L(x^i)$ pour un certain $i \in \{1, \ldots, n\}$ puis la décomposition de cet élément dans la base $(1, x, \ldots, x^{p-1})$. Pour le calcul des $L(x^i)$, cela revient à calculer les α_i définis dans la section 4.1.1 et à les évaluer en les points $0, \ldots, n-1$. D'après la preuve de la proposition 14, tout ceci peut se faire en $\mathcal{O}(n\,(A+B))$ opérations dans $\mathbb{F}_p$ (en négligeant les termes logarithmiques). Ensuite décomposer un élément de $\mathbb{F}_p(x)$ sur la base $(1, x, \ldots, x^{p-1})$ peut se faire à l'aide de divisions euclidiennes et d'utilisations de l'algorithme d'Euclide étendu pour un coût en $\mathcal{O}(\max(l,p)\,p)$ opérations dans $\mathbb{F}_p$. Nous avons besoin de faire p fois cette décomposition d'où un coût en $\mathcal{O}(\max(l,p)\,p^2)$. D'après la sous-section 1.2.2, l'étape 2 peut se faire en $\mathcal{O}(\max(l,p)^2\,p)$ opérations dans $\mathbb{F}_p$. Au final, nous obtenons le résultat annoncé. □

Par conséquent, une base de solutions polynomiales de L peut se calculer avec la même complexité : en effet, une fois que l'on a une base de solutions rationnelles, il ne reste plus qu'à multiplier les éléments de cette base par une constante convenable (par exemple le ppcm des dénominateurs des solutions rationnelles à la puissance p) pour obtenir une base de solutions polynomiales.

Remarque 15. *Les solutions polynomiales forment à la fois un $\overline{\mathbb{F}}_p$-espace vectoriel (qui est soit réduit à $\{0\}$ soit de dimension infinie) et un $\overline{\mathbb{F}}_p(x^p)$-espace vectoriel (ou $\overline{\mathbb{F}}_p[x^p]$-module). Dans ce chapitre, une base de solutions polynomiales signifie toujours une base du $\overline{\mathbb{F}}_p(x^p)$-espace vectoriel.*

Exemple 19. *Posons $p = 5$ et considérons l'opérateur différentiel*

$$L := (2\,x^{15} - x^6 + 2\,x^3 + 4\,x^9 + 1)\,x\,\partial^2 + (x^{15} + 3\,x^6 + x^9 - 2)\,\partial + 3\,x^5\,(3\,x^9 - 1).$$

La matrice de l'opérateur différentiel linéaire L dans la $\mathbb{F}_p(x^p)$-base $(1, x, x^2, x^3, x^4)$ de $\mathbb{F}_p(x)$ est :

```
> M:=evalm(matop(L,p));
```

$$M := \begin{bmatrix} 2\,x^5 & 3 & 0 & 2\,x^5 & 0 \\ 0 & 0 & 3 & 2\,x^{10} & 4\,x^5 \\ 0 & 0 & x^5 & 4\,x^{15} & 2\,x^{10} \\ 0 & 0 & 0 & 0 & 4 + 2\,x^{15} \\ 4\,x^{10} & x^5 & 4 & 0 & 2\,x^5 \end{bmatrix}$$

Nous calculons son noyau :

```
> N:=Nullspace(M) mod p;
```

$$N := \{[4,\, 0,\, x^{10},\, 1,\, 0],\ \left[\frac{1}{x^5},\, 1,\, 0,\, 0,\, 0\right]\}$$

et nous en déduisons qu'une base des solutions rationnelles de L est donnée par :

$$\{[4 + x^{10}\,x^2 + x^3],\ [\frac{1}{x^5} + x]\},$$

et par conséquent une base de solutions polynomiales est donnée par :

$$\{[x^{12} + x^3 + 4],\ [x^6 + 1]\}.$$

Compte tenu du résultat de complexité donné par le lemme 17, cette méthode pour calculer les solutions polynomiales est surtout intéressante pour des petits p.

Le but dans la suite de cette partie est de construire un algorithme modulaire pour calculer les solutions polynomiales d'un opérateur différentiel linéaire. Cet algorithme calcule des solutions polynomiales modulo plusieurs nombres premiers et reconstruit les

solutions en caractéristique zéro à l'aide d'une combinaison de l'algorithme des restes chinois et de la reconstruction rationnelle. Pour ce faire, nous avons besoin d'identifier parmi les solutions modulo plusieurs nombres premiers celles qui proviennent d'une même solution polynomiale en caractéristique zéro. Pour répondre à ce problème nous allons classer les solutions polynomiales mod p selon leur degré. La méthode décrite ci-dessus calcule les solutions polynomiales mod p sans aucune restriction sur leur degré et par conséquent, nous lui préfèrerons les méthodes décrites dans la sous-section suivante. Notons quand même que l'algorithme RatSols-Mod$p$, utilisé avec des petits premiers, donne un test de non-existence très intéressant (voir la sous-section 4.4.1).

4.3.2 Solutions polynomiales ayant un degré donné

Le lemme suivant montre pourquoi si l'on cherche à calculer les solutions polynomiales en caractéristique zéro, alors il n'est pas nécessaire de calculer toutes les solutions polynomiales modulo p : nous avons uniquement besoin de celles qui ont un degré inférieur ou égal à une borne sur le degré des solutions polynomiales en caractéristique zéro.

Lemme 18. *Soit $L \in \mathbb{Z}[x][\partial]$ et soit $y \in \mathbb{Q}[x]$ une solution polynomiale non-nulle de L. Alors il existe $q \in \mathbb{Q}$ tel que $\tilde{y} := q\,y \in \mathbb{Z}[x]$ soit réductible modulo p et $\tilde{y}[p]$ est une solution polynomiale non-nulle de $L[p]$.*

(ici $.[p]$ représente l'opérateur « réduction modulo p » : pour plus de détails, voir Chapitre 2, Section 2.1.3).

Démonstration. Il suffit de prendre q tel que $\tilde{y}$ possède un contenu (ici, le *contenu* d'un polynôme désigne le plus grand commun diviseur de ses coefficients) égal à 1. □

Par conséquent, une solution polynomiale y_p de $L[p]$ qui provient d'une solution polynomiale y de L en caractéristique zéro possède un degré inférieur ou égal à $\deg_x(y)$.

Il est clair que les algorithmes donnés dans la section 4.2 (ABP et Barkatou) peuvent être adaptés en caratéristique p. Nous devons cependant faire attention au choix de p : principalement, nous voulons éviter que des exposants distincts en caractéristique zéro donnent le même exposant modulo p. Pour pouvoir dérouler l'algorithme ABP en caractéristique p, nous devons choisir p tel que les $\alpha_{-A}(j)$ pour $j \in \{n, \ldots, N+A+B\}$ ne s'annulent pas : pour cette raison, nous prendrons toujours $p > N + A + B$. De même, dans l'algorithme de Barkatou, il est souhaitable de choisir $p > N$ où N est la plus grande racine entière positive de α_B. En effet, ceci évite que des racines de α_B (en caractéristique zéro) diffèrent d'un multiple de p et aient donc les mêmes réductions modulo p.

Remarque 16. *D'après le lemme 18 et la discussion qui précède cette sous-section, on ne doit, de toute façon, pas choisir un nombre premier p inférieur à N sinon les solutions modulo p risquent de ne pas avoir le même degré que les solutions en caractéristique zéro.*

Les études de complexité faites en caractéristique zéro ne dépendent pas de la caractéristique (voir [Bos03, 10.2]) de sorte que nous avons les mêmes résultats de complexité en caractéristique p. Cependant, en pratique, ce calcul est plus rapide en caractéristique p, principalement pour les deux raisons suivantes :

- les opérations se font dans $\mathbb{F}_p$,
- la taille des coefficients des systèmes linéaires que l'on a à résoudre est bornée (par p).

En comparant la complexité des propositions 14 et 15 à celle de la proposition 17, nous remarquons que la complexité de RatSols-Mod$p$ croît polynomialement en p alors que celle des algorithmes ABP et de Barkatou croît linéairement en N. Par conséquent et comme l'algorithme RatSols-Mod$p$ calcule des solutions polynomiales sans restriction sur le degré (voir l'explication à la fin de la sous-section précédente), nous préfèrerons par la suite utiliser les versions modulaires de ABP ou de Barkatou. Notons que si l'on veut calculer les solutions polynomiales modulo un $p < N$, alors nous pouvons toujours utiliser l'algorithme RatSols-Mod$p$.

4.4 Tester la non-existence

Dans cette partie nous ne cherchons pas à calculer les solutions polynomiales mais nous essayons juste de détecter (*a priori*) la non-existence de solution polynomiale non-nulle. Actuellement, les algorithmes classiques utilisent seulement le test basé sur la non-existence de racine entière positive du polynome indiciel de L en l'une de ses singularités. Cependant, dans la plupart des problèmes où on est amené à calculer les solutions polynomiales d'opérateurs auxiliaires, ces opérateurs auxiliaires passent ce test même s'ils n'ont pas de solution polynomiale. De plus, comme ces opérateurs auxiliaires possèdent des coefficients beaucoup plus gros que l'opérateur de départ, des tests modulaires s'avèrent très utiles en pratique.

4.4.1 Test rapide

Dans tout ce qui suit, pour ne pas avoir à discuter l'existence de la réduction $L[p]$ de L modulo p, nous supposons que nos opérateurs sont à coefficients dans $\mathbb{Z}[x]$.

Le lemme 18 et l'algorithme RatSols-Mod$p$ donnent un premier test modulaire (trivial) de non-existence de solution polynomiale non-nulle : si le déterminant de la matrice de l'opérateur linéaire $L[p]$ est non-nul, alors $L[p]$ n'a pas de solution rationnelle (et donc polynomiale) non-nulle et donc L n'a pas de solution polynomiale non-nulle.

Lemme 19. *Soit $L \in \mathbb{Z}[x][\partial]$. Soit l une borne sur le degré (en x) des coefficients de L. Tester la non existence de solution polynomiale avec le test modulaire précédent peut se faire en $\mathcal{O}(n\,(A+B)\,\max(l,p)^2\,p^\omega\,\max(\frac{l}{p},1))$ opérations dans $\mathbb{F}_p$, où ω est un réel tel que le produit de deux matrices dans $\mathbb{M}_n(\mathcal{F})$, avec $\mathcal{F}$ corps quelconque, puisse se faire en $\mathcal{O}(n^\omega)$ opérations dans $\mathcal{F}$.*

Démonstration. D'après la preuve du lemme 17, construire la matrice de l'opérateur linéaire $L[p]$ peut se faire en $\mathcal{O}(n\,(A+B)\,\max(l,p)^2\,p)$ opérations dans $\mathbb{F}_p$. Maintenant,

d'après [St03], calculer le déterminant d'une matrice $p \times p$ ayant des entrées de degré borné par d peut se faire en $\mathcal{O}(p^\omega\, d)$ opérations de corps (en négligeant les termes logarithmiques) où ω est l'exposant de la multiplication de matrices. Or, dans le chapitre 1, nous avons vu dans la section 1.2 que le degré des entrées de la matrice était borné par $l/p + 1$ d'où le résultat. □

La complexité de ce test dépend donc fortement de p et en pratique, ce test est surtout intéressant si on l'utilise avec des petits nombres premiers (*e.g.*, $p = 2$, 3, 5). Remarquons qu'un intérêt de ce test est qu'il ne dépend pas d'une borne N sur le degré des solutions polynomiales ; ceci est cohérent avec le fait que cette méthode calcule des solutions polynomiales sans se soucier de leur degré.

Exemple 20. *Considérons l'opérateur différentiel linéaire suivant :*

$$\begin{aligned}L = {} & (x^2+x+8)^2\,\partial^4+(x^2+x+8)\,(2\,x^8-5814\,x-5331)\,\partial^3+(x^{16}+256\,x^7+23261828\,x\\ & +10595524-8696\,x^9+12684698\,x^2-7963\,x^8)\,\partial^2+(-25039\,x^{12}+359649\,x^{10}-1163014557\\ & +11410642\,x^5-621906702\,x^3-159337\,x^{11}+23047\,x^{13}+6957787\,x^7+8819879\,x^8\\ & +249\,x^{14}+530621566\,x^4-2888\,x^{15}-67752682\,x^6-3710072636\,x-3623065826\,x^2\\ & +915047\,x^9)\partial+2086568\,x^{14}-35187260057\,x^5-359808\,x^{13}-2959136216\,x^8\\ & +288366200961\,x^2-828786528\,x^9+5642839360\,x^7+12468795209\,x^6+47582358547\\ & +199846009415\,x-46102704943\,x^4+39075760\,x^{11}+130179393143\,x^3+360154648\,x^{10}\\ & -33298936\,x^{12}.\end{aligned}$$

Pour $p = 3$, calculons la matrice de l'opérateur $L[p]$ dans la base $(1, x, x^2)$ de $\mathbb{F}_p(x)$ sur $\mathbb{F}_p(x^p)$.

```
> p:=3: M:=evalm(matop(L,p));
```

$$M := \begin{bmatrix} 1+2\,x^3+2\,x^6+2\,x^{12} & 0 & 2+x^3+x^6+x^9+2\,x^{12} \\ 2+2\,x^3+x^6+x^9 & 2 & 1+x^6+2\,x^9+2\,x^{12} \\ 2\,(2+2\,x^3+2\,x^6+x^9)\,x^3 & 0 & 1+x^3+2\,x^6+x^{12} \end{bmatrix}.$$

Calculons son déterminant :

```
> Det(M) mod p;
```

$$2\,x^{24}+x^{21}+2\,x^{18}+2\,x^{15}+x^9+2\,x^3+2.$$

Par conséquent L n'a pas de solution polynomiale non-nulle. Remarquons que pour cet exemple comme l'ensemble des degrés possibles pour des solutions polynomiales de L est $\mathcal{I}(L, \infty) = \{1441, 1448\}$, les algorithmes usuels mettent beaucoup de temps pour arriver à la conclusion que L n'a pas de solution polynomiale non-nulle.

Il existe cependant des cas où l'opérateur passe ce test pour tout p bien qu'il ne possède pas de solution polynomiale non-nulle en caractéristique zéro : les deux exemples suivants illustrent ce fait.

Exemple 21. *Considérons l'opérateur différentiel*

$$L = (x+1)\,\partial^2 + (x^2+x-10)\,\partial - 9\,x + 1.$$

Cet opérateur n'a pas de solution polynomiale non-nulle en caractéristique zéro mais on peut montrer que pour tout p, $L[p]$ possède une solution polynomiale non-nulle : en effet, nous avons $L = ((x+1)\,\partial - 10)\,(\partial + x) := L_1\,L_2$ où L_1 possède la solution polynomiale $Q := (x+1)^{10}$ et L_2 n'a pas de solution rationnelle non-nulle. Maintenant, pour tout nombre premier p, L_2 induit une bijection $L_2[p]$ de $\mathbb{F}_p(x)$ dans lui-même (car il n'a pas de solution rationnelle) : soit T l'inverse de cette bijection. Alors, pour tout premier p, $T(Q[p])$ est une solution rationnelle de $L[p]$ et notre test échoue pour tout p.

Exemple 22. *Considérons l'opérateur d'Euler homogène*

$$L = x^3\,\partial^2 + x\,(x+1)\,\partial - 1.$$

Cet opérateur n'a pas de solution polynomiale non-nulle en caractéristique zéro. Par contre, il possède $\hat{y}(x) = \sum_{i=0}^{\infty}(-1)^i\,i!\,x^{i+1}$ comme solution formelle (l'autre solution formelle étant $\exp(1/x)$) et par conséquent, pour tout p, $L[p]$ possède la solution polynomiale $\hat{y}[p](x) = \sum_{i=0}^{p-1}(-1)^i\,i!\,x^{i+1}$.

4.4.2 Un nouveau test modulaire

Lorsque la borne sur le degré N est grande (et que les tests décrits précédemment échouent), alors nous voulons commencer à calculer des solutions polynomiales seulement si celles-ci ont de grandes chances d'exister. Par conséquent, dans ce cas, nous pouvons nous autoriser à passer plus de temps pour essayer de détecter la non-existence de solutions polynomiales.

Soit $L \in \mathbb{Z}[x][\partial]$ et notons $\mathcal{V}_N$ l'espace vectoriel des solutions polynomiales de L de degré inférieur ou égal à N. Une première borne *a priori* sur la dimension de $\mathcal{V}_N$ est donnée par $\#\mathcal{I}(L,\infty)$ (voir la sous-section 4.1.2).

Comme nous l'avons vu dans l'exemple 21, l'opérateur différentiel réduit $L[p]$ peut avoir des solutions polynomiales non-nulles pour tout p même lorsque L ne possède pas de solution polynomiale non-nulle en caractéristique zéro. Cependant, en prenant en compte une borne sur le degré, nous avons :

Théorème 4. *Soit $L \in \mathbb{Z}[x][\partial]$ et soit N une borne sur le degré d'éventuelles solutions polynomiales de L. Alors pour presque tout p, la dimension de l'espace des solutions polynomiales de degré inférieur ou égal à N de L et $L[p]$ est la même.*

Démonstration. Pour calculer la dimension de $\mathcal{V}_N$, on peut écrire une éventuelle solution polynomiale de L sous la forme $\sum_{i=0}^{N} y_i\,x^i$ et écrire le système linéaire pour les inconnues $y_i \in \mathbb{Q}$ (voir Section 4.2.1). La matrice M de ce système est alors de dimension $(N+A+$

$B+1)\times(N+1)$ où A et B sont définis dans (4.1). La dimension de $\mathcal{V}_N$ en caractéristique zéro est égale à $N+1-\text{rang}(M)$. Maintenant, le même procédé peut être appliqué en caractéristique p à $L[p]$ et la matrice du système obtenu est exactement $M[p]$. Par conséquent, la dimension de l'espace des solutions polynomiales de degré inférieur ou égal à N de $L[p]$ est $N+1-\text{rang}(M[p])$. Nous concluons en remarquant que pour presque tout p, $\text{rang}(M[p]) = \text{rang}(M)$; en effet, il suffit de supposer que l'élimination de Gauss sur les matrices M et $M[p]$ se déroule de la même façon ce qui n'exclut qu'un nombre fini de premiers. □

La preuve de ce théorème nous indique quels vont être les *mauvais premiers* c'est-à-dire ceux pour lesquels $L[p]$ possèdera une (ou plusieurs) solution(s) parasite(s) : avec les notations de la preuve, ceux sont les p tels que $\text{rang}(M[p]) < \text{rang}(M)$. Cependant, si l'on cherche à calculer ces nombres premiers p, il nous faut calculer et factoriser le déterminant principal de M et le coût de cette opération est à peu près équivalent à celui de la résolution du système $MY = 0$ c'est-à-dire à celui du calcul des solutions polynomiales. Par conséquent en pratique nous ne calculerons pas ces mauvais premiers (voir la sous-section 4.5.2) mais nous utiliserons plutôt une stratégie reposant sur l'argument portant sur le degré des solutions parasites donné par le corollaire suivant.

Corollaire 3. *Soit $L \in \mathbb{Z}[x][\partial]$ et soit N une borne sur le degré d'éventuelles solutions polynomiales de L. Il n'existe qu'un nombre fini de nombres premiers p tels que $L[p]$ possède une solution polynomiale de degré $\leq N$ ne provenant pas de caractéristique zéro (au sens du lemme 18).*

Démonstration. Raisonnons par l'absurde et supposons que pour une infinité de p, $L[p]$ possède une solution polynomiale non-nulle ne provenant pas d'une solution de L et ayant un degré inférieur ou égal à N. Comme cette solution ne provient pas d'une solution de L, pour une infinité de p, la dimension de l'espace des solutions polynomiales de degré inférieur ou égal à N de $L[p]$ sera supérieure à celle de L ce qui est en contradiction avec le théorème précédent. □

Ces résultats théoriquement intéressants mènent à un test de non-existence de solution polynomiale non-nulle :

1. calculer une borne N sur le degré d'éventuelles solutions polynomiales non-nulles de L en caractéristique zéro,
2. choisir un premier $p > N + A + B$,
3. calculer le rang r_p de $M[p]$ (voir la preuve du théorème 4 ci-dessus). Si $r_p = N+1$, alors $L[p]$ n'a pas de solution polynomiale non-nulle de degré $\leq N$ et donc L n'a pas de solution polynomiale non-nulle.

L'algorithme de [MS98] calcule le rang de $M[p]$ en $\mathcal{O}((N+A+B)\,N^{\omega-1})$ opérations dans $\mathbb{F}_p$ où N est une borne sur le degré d'éventuelles solutions polynomiales, A et B sont définis dans (4.1) et ω est l'exposant de l'algèbre linéaire : cette complexité est équivalente à celle pour résoudre le système linéaire. Ceci limite donc l'intérêt de

ce test en pratique. Cependant, récemment, dans [SSV04], les auteurs montrent comment calculer le rang d'une matrice à coefficients dans $\mathbb{Z}$ d'une manière plus efficace : ces résultats sont encourageants et nous espérons qu'ils pourront être généralisés à la caractéristique p (actuellement cette méthode ne s'applique pas en caractéristique p).

Exemple 23. *Revenons à l'exemple 21. Pour cet exemple le test modulaire de la sous-section précédente échoue. En regardant d'un peu plus près, on voit que* $\mathcal{I}(L, \infty) = \{9\}$ *et donc* $N = 9$ *est le degré d'une éventuelle solution polynomiale. De plus, nous avons* $A = 2$ *et* $B = 1$ *de sorte que nous choisissons* $p > 12$*. Si nous appliquons le test ci-dessus à* L*, alors nous détecterons la non-existence de solution polynomiale non-nulle sauf si le* p *choisi à l'étape 2 est* $p = 19$*. En effet, pour tout nombre premier* p *strictement supérieur à* 12 *sauf pour* $p = 19$*, nous avons* $r_p = N + 1$ *puisque, pour cet exemple,* $p = 19$ *est l'unique mauvais premier supérieur à* 12 *; ceci vient du fait que le déterminant principal de la matrice* M *est égal à* $2^{14} \cdot 3^4 \cdot 5^2 \cdot 7 \cdot 19$*.*

La méthode suggérée par le test donné ici donne aussi une heuristique pour calculer la dimension de l'espace des solutions polynomiales en caractéristique zéro : en effet, avec les notations précédentes, pour presque tout p, cette dimension est égale à $N+1-\mathrm{rang}(M[p])$. Cependant, dès que N est « grand », il est plus judicieux de calculer la dimension de l'espace des solutions polynomiales modulo p en calculant une base de solutions polynomiales grâce à la version modulaire de ABP ou Barkatou modulo p ; en effet au lieu d'être en $\mathcal{O}(N^\omega)$, la complexité est alors linéaire en N.
Le fait de disposer d'une heuristique pour calculer la dimension de l'espace des solutions polynomiales en caractéristique zéro est particulièrement intéressant pour certains problèmes de théorie de Galois différentielle où cette dimension donne des informations sur le groupe cherché. Par exemple, les solutions rationnelles de puissances symétriques d'opérateurs différentiels linéaires induisent des espaces d'invariants des groupes de Galois différentiels, et on discrimine les groupes entre eux en comparant ces dimensions (voir [HRUW98, Algorithm 4.1], [Ul03, Theorem 3.2 and Algorithm 3.4] ou encore [Cor01, CSTU02]).

4.5 Reconstruire les solutions polynomiales en caractéristique zéro

Dans ce qui suit, notre but est de calculer une base de solutions polynomiales de $L[p]$ pour plusieurs premiers p et de les combiner pour obtenir une base de solutions polynomiales de L. Pour ce faire, pour chaque élément y d'une base de solutions polynomiales de L, nous devons être capable de retrouver $y[p]$ parmi les éléments d'une base de solutions polynomiales de $L[p]$. De plus, nous devons nous assurer que les bases modulaires ne contiennent pas de « solution parasite », *i. e.*, des solutions qui ne proviennent pas de solutions en caractéristique zéro. Par conséquent, nous devons calculer des bases de solutions polynomiales sous une forme canonique et exclure certains mauvais nombres premiers.

4.5.1 Bases échelonnées

Dans cette sous-section, si aucune autre condition n'est précisée, $\mathbb{K}$ représente soit $\mathbb{Q}$ soit $\mathbb{F}_p$.

Définition 26. *Soit $L \in \mathbb{K}[x][\partial]$ et soit $\mathcal{B} = \{y^{[1]}, \dots, y^{[r]}\}$ avec $y_i \in \mathbb{K}[x]$ une base de solutions polynomiales de L. Pour $i \in \{1, \dots, r\}$, on note $y^{[i]} = \sum_{j=0}^{d_i} y_j^{[i]} x^j$ avec les $d_i \in \mathbb{N}$ tels que $d_1 > \cdots > d_r$, les $y_j^{[i]} \in \mathbb{K}$ et pour tout $i \in \{1, \dots, r\}$, $y_{d_i}^{[i]} \neq 0$.*
$\mathcal{B}$ est une base échelonnée *si pour tout $i \in \{1, \dots, r\}$, $y_{d_i}^{[i]} = 1$ et, pour tout i, $j \in \{1, \dots, r\}$ avec $i \neq j$, $y_{d_j}^{[i]} = 0$.*

Ces bases échelonnées apparaissent d'une manière tout à fait naturelle et sont les bases calculées par la plupart des algorithmes classiques.

Étant donné une base quelconque $\{y^{[1]}, \dots, y^{[r]}\}$ de solutions polynomiales, une base échelonnée peut se calculer en faisant de l'élimination de Gauss sur les vecteurs formés des coefficients des $y^{[i]}$.

Nous avons de plus le résultat d'unicité (classique et très simple) suivant :

Lemme 20. *Étant donné $L \in \mathbb{K}[x][\partial]$, il existe une unique base échelonnée de solutions polynomiales de L.*

Le théorème suivant fait le lien entre la base échelonnée de solutions polynomiales d'un opérateur différentiel L en caractéristique zéro et celle de sa réduction $L[p]$ modulo p :

Théorème 5. *Soit $L \in \mathbb{Z}[x][\partial]$ et soit N une borne sur le degré d'éventuelles solutions polynomiales de L. Soit $\mathcal{B} = \{y^{[1]}, \dots, y^{[r]}\}$ la base échelonnée de solutions polynomiales de degré $\leq N$ de L. Alors, pour presque tout p, $\mathcal{B}[p] := \{y^{[1]}[p], \dots, y^{[r]}[p]\}$ est la base échelonnée, notée $\mathcal{B}_p$, de solutions polynomiales de degré $\leq N$ de $L[p]$.*

Démonstration. Tout d'abord, d'après le théorème 4, pour presque tout p, $\mathcal{B}[p]$ et $\mathcal{B}_p$ ont le même nombre d'éléments. Maintenant, d'après le lemme 20, il y a une unique base échelonnée de solutions polynomiales de degré $\leq N$ de $L[p]$ et donc, il nous suffit de montrer que, pour presque tout p, $\mathcal{B}[p]$ est une base échelonnée. Or, comme les éléments de $\mathcal{B}$ sont supposés unitaires, cette dernière affirmation est vraie pour tout p n'apparaissant pas dans les dénominateurs des coefficients. Par conséquent, dès que $\mathcal{B}[p]$ est bien définie, c'est encore une base échelonnée. □

Remarque 17. *Ce théorème et le théorème 4 ne restent pas vrais si l'on omet la condition sur le degré des solutions polynomiales considérées qui est supposé $\leq N$. En effet, un opérateur différentiel linéaire peut posséder des solutions polynomiales non-nulles mod p pour tout p et ne pas avoir de solutions polynomiales non-nulles en caractéristique zéro (voir l'exemple 21).*

Avec les notations du théorème précédent, il est possible que $\mathcal{B}[p] \neq \mathcal{B}_p$:

Exemple 24. *Considérons l'opérateur différentiel*

$$L \;=\; 3\,x^2\,\partial^2 - 19\,x\,\partial + 7.$$

On a alors $\mathcal{B} = \{x^7\}$*, et donc* $\mathcal{B}[11] = \{x^7\}$ *mais* $\mathcal{B}_{11} = \{x^7, x^4\}$.

Comme nous supposons toujours $p \geq N$, les degrés des éléments de $\mathcal{B}$ et $\mathcal{B}[p]$ définis au théorème 5 doivent être les mêmes. Donc, en pratique, lorsque l'on dispose de deux bases modulaires $\mathcal{B}_{p_1}$ et $\mathcal{B}_{p_2}$ pour deux premiers distincts p_1 et p_2, nous devons tout d'abord vérifier que les degrés des éléments des deux bases sont les mêmes et qu'ils appartiennent à $\mathcal{I}(L, \infty)$. Si cela n'est pas vérifié, alors l'un des deux premiers est mauvais.

4.5.2 Nombres premiers pertinents

Comme nous l'avons déjà fait remarquer, pour utiliser les algorithmes ABP et Barkatou en caractéristique p, nous devons supposer que les nombres premiers p que l'on prend sont supérieurs à $N + A + B$, où N est une borne sur le degré d'éventuelles solutions polynomiales de L et les entiers A et B sont définis par (4.1).

Comme nous sommes seulement intéressés par les solutions polynomiales mod p qui correspondent à des solutions polynomiales de degré $\leq N$ en caractéristique zéro, nous voulons essayer d'éviter l'apparition de nouvelles solutions « parasites » mod p. Le théorème 4 montre que cette apparition de solutions « parasites » ne peut survenir que pour un nombre fini de nombres premiers. Comme nous l'avons remarqué dans le chapitre 2, cette observation ne peut pas être utile en pratique si nous ne pouvons pas exhiber les mauvais premiers ce qui est malheureusement le cas ici ; rappelons que le coût du calcul des mauvais premiers est équivalent à celui du calcul d'une base de solutions polynomiales en caractéristique zéro (voir l'explication suivant le théorème 4). Cependant, quelques mauvais premiers peuvent tout de même être retirés à l'avance à moindre coût.

Proposition 16. *Soit* $L = \sum_{i=0}^{n} a_i\,\delta^i \in \mathbb{Z}[x][\delta]$ *où* $\delta = x\,\partial$ *et soit* $\alpha(L, \infty)$ *son polynôme indiciel à l'infini. Soit* p *un nombre premier tel que : pour tout* $i \in \{0, \ldots, n\}$, $\mathrm{lcoeff}(a_i, x) \neq 0 \bmod p$ *(où* $\mathrm{lcoeff}(f, x)$ *désigne le coefficient dominant d'un polynôme* $f \in \mathbb{K}[x]$*). Alors le polynôme indiciel* $\alpha(L[p], \infty)$ *de* $L[p]$ *à l'infini est égal à* $\alpha(L, \infty)[p]$*, la réduction de* $\alpha(L, \infty)$ *modulo* p*.*

Démonstration. Ceci provient du fait que, lorsque L est écrit en la dérivation d'Euler δ, les coefficients dominants de ses coefficients a_i sont exactement les coefficients du polynôme indiciel de L à l'infini. □

Soit $L = \sum_{i=0}^{n} a_i\,\partial^i \in \mathbb{Z}[x][\partial]$ et p un premier supérieur à $N + A + B$, où A et B sont définis dans (4.1) et N est une borne sur le degré d'éventuelles solutions polynomiales de L. Même lorsque pour tout $i \in \{0, \ldots, n\}$, $\mathrm{lcoeff}(a_i, x) \neq 0 \bmod p$, les degrés des élements d'une base échelonnée de solutions polynomiales de $L[p]$ peuvent ne pas appartenir à $\mathcal{I}(L, \infty)$. En effet, une racine r rationnelle (mais non entière) du polynôme indiciel de L à l'infini peut satisfaire $r[p] \leq N$ ($r[p] \in \mathbb{F}_p$ vu comme un élément de $\mathbb{N}$ qui est $\leq p - 1$) et conduire à une nouvelle solution polynomiale mod p.

Exemple 25. *Considérons l'opérateur de l'exemple 24. Il admet* $3(N-7)\,(N-1/3)$ *comme polynôme indiciel à l'infini. Une borne sur le degré d'éventuelles solutions polynomiales de* L *est donc* $N = 7$*. De plus, nous avons* $A = B = 0$ *de telle sorte que l'on peut prendre n'importe quel premier strictement plus grand que* 7 *excepté* $p = 19$ *qui modifie le polynôme indiciel à l'infini (voir Proposition 16). Maintenant, comme la réduction de* $1/3$ *modulo* $p = 11$ *(resp.* $p = 17$*) est égale à* $4 \leq N$ *(resp.* $6 \leq N$*), nous n'allons pas considérer les premiers* $p = 11$ *et* $p = 17$*. En effet, modulo* $p = 11$ *et* $p = 17$*,* $L[p]$ *possède une solution polynomiale de degré* $\leq N$ *qui ne vient pas d'une solution polynomiale de* L *en caractéristique zéro. Notons que pour tout premier* $p > 7$*, et* $p \notin \{11, 17\}$*,* $1/3 \bmod p > N$*.*

Définition 27. *Soit* $L = \sum_{i=0}^{n} a_i\,\partial^i \in \mathbb{Z}[x][\partial]$ *et* N *une borne sur le degré d'éventuelles solutions polynomiales de* L*. On dit qu'un nombre premier* p *est* pertinent pour L *si il est strictement plus grand que* $N + A + B$ *et si il satisfait de plus les deux conditions suivantes :*

- *si l'on note* $\tilde{L} = \sum_{i=0}^{n} \tilde{a}_i\,\delta^i$ *l'opérateur* L *écrit en la dérivation d'Euler* $\delta = x\,\partial$*, alors, pour tout* $i \in \{1, \ldots, n\}$*,* $\mathrm{lcoeff}(\tilde{a}_i, x) \neq 0 \bmod p$*,*
- *il n'y a pas de racine rationnelle du polynôme indiciel de* L *à l'infini dont la réduction modulo* p *est un élément de* $\mathbb{F}_p$ *inférieur ou égal à* N*.*

Remarque 18. *En pratique, il est facile de tester si un nombre premier* p *est pertinent. Nous pouvons de plus exhiber un ensemble contenant tous les premiers non-pertinents. Ces nombres premiers sont* pertinents *plus que* bons *car il se peut que modulo un premier* p *pertinent,* $L[p]$ *possède des solutions parasites : dans l'exemple 23,* $p = 19$ *est pertinent mais* $L[p]$ *possède une solution parasite. Cependant, comme nous l'avons déjà remarqué (voir l'explication suivant le théorème 4), il serait trop coûteux de calculer les mauvais nombres premiers.*

La stratégie employée dans notre algorithme final (voir la sous-section 4.6.2) nous oblige à disposer d'une infinité de nombres premiers : comme nous n'utiliserons que des premiers pertinents, nous devons montrer qu'il en existe une infinité :

Proposition 17. *Étant donné* $L \in \mathbb{Z}[x][\partial]$*, il existe une infinité de nombres premiers pertinents pour* L*.*

Démonstration. Il nous suffit de montrer qu'étant donné $N \in \mathbb{N}$ et $q \in \mathbb{Q}$, alors pour p assez grand, $q[p] > N$ [2]. Soit donc $q = \frac{n}{d} \in \mathbb{Q}$ avec n, $d \in \mathbb{Z}^*$ et $d \notin \mathbb{Z}^\times = \{-1, 1\}$. Soit p un premier tel que $\mathrm{pgcd}(d, p) = 1$ (pour p assez grand ceci est toujours vérifié). Le théorème de Bézout nous affirme alors qu'il existe a, $b \in \mathbb{Z}$ tels que $a\,d + b\,p = 1$, d'où $(a\,n)\,d + (b\,n)\,p = n$. De plus, la division euclidienne de $a\,n$ par p nous fournit s, r tels que $a\,n = s\,p + r$ avec $0 \leq r < p$. On obtient donc $r\,d + b'\,p = n$ avec $b' = s\,d + b\,n$, d'où $r = \frac{n}{d} - \frac{b'}{d}\,p$. Or par définition $r = q[p]$ donc en prenant les valeurs absolues et en utilisant l'inégalité triangulaire, on obtient $|q[p]| \geq |\frac{b'}{d}\,p| - |q|$. Maintenant comme

[2] Pour clarifier cette inégalité, nous devons définir précisément $q[p]$: le rationnel q, vu comme un élément de l'anneau $\mathbb{Z}_p$ des entiers p-adiques est envoyé sur un élément de $\mathbb{F}_p = \mathbb{Z}_p/(p)$ que l'on plonge ensuite dans $\{0, \ldots, p-1\}$ pour obtenir $q[p]$.

$b' \neq 0$ (sinon on aurait $r = \frac{n}{d} \in \mathbb{Z}$ ce qui impliquerait $d \in \mathbb{Z}^\times$), en faisant tendre p vers $+\infty$, on obtient $\lim_{p\to+\infty} |q[p]| = +\infty$ d'où le résultat. □

Remarque 19. *La preuve ci-dessus peut être généralisée au cas où q appartient à un corps de nombre quelconque : plus précisément, on peut montrer que si K est une extension finie de $\mathbb{Q}$, si $q = \frac{n}{d} \in K$ avec $n \in \mathcal{O}_K$, l'anneau des entiers de K, $d \in \mathbb{Z}$ et $q \notin \{0, \ldots, p-1\}$, alors étant donné $N \in \mathbb{N}$, pour p assez grand $q[p] > N$. Cela nous permettrait d'affiner la deuxième condition sur les nombres premiers pertinents qui deviendrait :*

- *il n'y a pas de racine non-entière du polynôme indiciel de L à l'infini dont la réduction modulo p est un élément de $\mathbb{F}_p$ inférieur ou égal à N.*

Dans notre algorithme, si nous avons déjà calculé des bases échelonnées de solutions polynomiales modulo plusieurs premiers pertinents et si ces bases n'ont pas le même nombre d'éléments, alors nous jetons celles qui ont le plus d'éléments : en effet, la théorie nous assure que celles-ci contiennent trop d'éléments (*e.g.*, des solutions parasites). Les bases que l'on garde peuvent très bien être jetées si l'on rajoute des calculs modulo d'autres premiers pertinents. Cependant, nous pouvons assurer (voir Théorème 4) que ceci ne peut arriver qu'un nombre fini de fois et donc, pour p assez grand, toutes les bases calculées auront le même nombre d'éléments.

4.5.3 Comment combiner les solutions polynomiales modulo plusieurs premiers ?

Soit L un opérateur différentiel linéaire et N une borne sur le degré d'éventuelles solutions polynomiales de L. Soit $p_1, \ldots, p_m$ des nombres premiers pertinents pour L et $\mathcal{B}_{p_1}, \ldots, \mathcal{B}_{p_m}$ les bases échelonnées de solutions polynomiales de degré $\leq N$ de $L[p_1], \ldots, L[p_m]$. Supposons qu'elles possèdent toutes le même nombre d'éléments (voir la discussion à la fin de la sous-section précédente). Pour i dans $\{1, \ldots, m\}$, on pose $\mathcal{B}_{p_i} := \{y_{p_i}^{[1]}, \ldots, y_{p_i}^{[s]}\}$. En considérant leur degré, nous pouvons relier un à un les éléments des $\mathcal{B}_{p_i}$: supposons alors que pour tout i, j dans $\{1, \ldots, m\}$ et k dans $\{1, \ldots, s\}$, $y_{p_i}^{[k]}$ et $y_{p_j}^{[k]}$ aient le même degré. Pour k dans $\{1, \ldots, s\}$, nous pouvons utiliser l'algorithme des restes chinois (voir [GG99, Section 5.4]) et la reconstruction rationnelle (voir [GG99, Section 5.10] ou [CE95, Mo04]) pour trouver un polynôme $y^{[k]} \in \mathbb{Q}[x]$ dont les réductions modulo $p_1, \ldots, p_m$ sont respectivement égales à $y_{p_1}^{[k]}, \ldots, y_{p_m}^{[k]}$. Voici brièvement comment on procède :

Algorithme CombPolySols

Entrée : des nombres premiers $p_1, \ldots, p_m$, $y_{p_1}^{[k]} \in \mathcal{B}_{p_1}, \ldots, y_{p_m}^{[k]} \in \mathcal{B}_{p_m}$ ayant le même degré.
Sortie : $y^{[k]} \in \mathbb{Q}[x]$ tel que $y^{[k]} \equiv y_{p_i}^{[k]} \mod p_i$ pour $i = 1, \ldots, m$.

1. Si $m > 1$, alors utiliser l'algorithme des restes chinois pour calculer $f \in \mathbb{Z}[x]$ tel que $f \equiv y_{p_i}^{[k]} \mod p_i$ pour $i \in \{1, \ldots, m\}$. Sinon, poser $f := y_{p_1}^{[k]}$.
2. Appliquer la reconstruction rationnelle aux coefficients de f.

Si nous avons calculé les solutions polynomiales modulo un seul premier ($m = 1$), alors l'algorithme des restes chinois est inutile.
On peut tester si $f \in \mathbb{Z}[x]$ vérifie $L(f) = 0$ avant de faire la reconstruction rationnelle. Notons $P := \prod_{i=1}^{m} p_i$. Appliquer la reconstruction rationnelle aux coefficients f_i de f signifie, pour chaque i, chercher n_i et d_i tels que $\frac{n_i}{d_i} = f_i \mod P$, $\gcd(d_i, P) = 1$, $|n_i| < \sqrt{P/2}$ et $0 \leq d_i \leq \sqrt{P/2}$.
Cette reconstruction rationnelle peut échouer (par exemple si P n'est pas assez grand) : dans ce cas, nous calculons des solutions polynomiales modulo plus de nombres premiers et nous recommençons.

Remarque 20. *Si l'on note b la borne donnée par la proposition 18 (voir la sous-section 4.6.1) sur la taille du numérateur et du dénominateur des solutions polynomiales de L, alors on pourrait utiliser b à la place de $\sqrt{P/2}$ comme borne sur la taille des n_i et d_i dans ce qui précède : cependant, comme nous le verrons par la suite cette borne est en général beaucoup plus grande que la taille des coefficients de sorte qu'il est plus judicieux d'utiliser $\sqrt{P/2}$. Notons que dès que $\sqrt{P/2}$ est supérieur à b nous pouvons être sûr que s'il existe des solutions polynomiales en caractéristique zéro, alors nous disposons de suffisamment de calculs modulaires pour les reconstruire.*

Lemme 21. *Soit $P := \prod_{i=1}^{m} p_i$. La complexité de l'algorithme* CombPolySols *donné ci-dessus est en $\mathcal{O}(N \log(P)^2)$ opérations de mots.*

À partir de maintenant, nous allons prendre en compte la taille (machine) des entiers sur lesquels nous travaillons et donc donner nos complexités en « opérations de mots ». Cela nous permettra de pouvoir comparer la complexité de notre algorithme modulaire à celle des algorithmes de la section 4.2.

Le résultat donné dans ce lemme est la complexité pour reconstruire une seule éventuelle solution polynomiale en caractéristique zéro de sorte que ceci doit être fait au plus n fois (puisque l'ordre n de L est aussi une borne sur le nombre de solutions polynomiales de L linéairement indépendantes).

Démonstration. Soit $P := \prod_{i=1}^{m} p_i$. D'après [GG99, Theorem 5.8], la première étape de notre algorithme peut se faire en $\mathcal{O}(N \log(P)^2)$ opérations de mots (note : si N est une borne sur le degré, alors N est aussi une borne sur le nombre de coefficients des $y_{p_i}^{[k]}$). La reconstruction rationnelle repose sur l'algorithme d'Euclide étendu (voir [GG99, Sections 5.10 and 11.1] ou bien [Mo04]) et peut elle aussi se faire en $\mathcal{O}(N \log(P)^2)$ d'où le résultat. □

Exemple 26. *Considérons l'opérateur différentiel linéaire*

$$L = (-21x^4 + 6x - 489x^2)\,\partial^2 + (56x^3 + 2 + 326x)\,\partial - 42x^2 - 326.$$

Nous avons $\mathcal{I}(L,\infty)=\{3\}$*, et donc une borne sur le degré d'éventuelles solutions polynomiales de* L *est* $N=3$*. Nous ne considérons ici que des nombres premiers strictement plus grands que* $N+A+B=3+1+2=6$ *excepté* $p=7$ *qui n'est pas pertinent car tous les* $\operatorname{lcoeff}(a_i,x)$ *se réduisent à* 0 *mod* 7*.*
Nous obtenons les solutions suivantes : $x^3+9\,x+1 \mod 11$*,* $x^3+7\,x+1 \mod 13$*. L'algorithme des restes chinois appliqué à ces solutions produit le polynôme* $x^3+20\,x+1$*. Nous testons si ce polynôme* y *satisfait la condition* $L(y)=0$*. Comme il ne la satisfait pas, nous lançons la reconstruction rationnelle qui nous renvoie le polynôme* $x^3-\frac{3}{7}\,x+1$ *qui, lui non plus, ne satisfait pas* $L(y)=0$*. Par conséquent, nous cherchons les solutions polynomiales modulo le nombre premier pertinent suivant qui est ici* $p=17$*. Nous obtenons la solution polynomiale* $x^3+10\,x+1 \mod 17$*. Nous appliquons l'algorithme des restes chinois qui nous donne le polynôme* $x^3+163\,x+1$*. Ce polynôme satisfait* $L(y)=0$ *et donc, la base échelonnée de solutions polynomiales de* L *est* $\{x^3+163\,x+1\}$*.*

Nous pourrions aussi reconstruire les solutions polynomiales en caractéristique zéro en calculant les solutions polynomiales de L modulo p, p^2, ... Comme cette méthode est plus efficace dans le cas de la résolution de système linéaire, il se peut qu'ici aussi cela soit plus efficace. Cependant, cette approche n'utilisant qu'un seul nombre premier p, le choix de ce nombre premier serait alors crucial ce qui constituerait un réel problème puisque qu'actuellement nous ne pouvons pas exhiber un « bon » premier à moindre coût : en effet, le calcul des mauvais premiers a un coût équivalent au calcul d'une base de solutions polynomiales en caractéristique zéro (voir l'explication suivant le théorème 4). De plus il faudrait aussi voir comment calculer les solutions polynomiales d'un opérateur différentiel linéaire modulo des puissances de p : ceci devrait pouvoir se faire avec l'algorithme ABP ou celui de Barkatou puisque ceux-ci fonctionnent aussi lorsque l'équation est non-homogène (voir [ABP95, Ba97]). Notons aussi qu'un avantage de notre procédé utilisant plusieurs nombres premiers, est que nous avons ainsi plus de chances de détecter rapidement la non-existence de solutions polynomiales non-nulles.

4.6 Algorithme modulaire

4.6.1 Borne sur la taille des coefficients

En résumé, la démarche que nous allons suivre dans notre algorithme modulaire est la suivante : nous calculons des bases de solutions polynomiales de L modulo plusieurs nombres premiers pertinents, puis, nous reconstruisons (à l'aide de l'algorithme CombPolySols) à partir de ces solutions polynomiales modulaires des polynômes à coefficients dans $\mathbb{Q}$. Ensuite, nous testons si ces polynômes sont bien des solutions en caractéristique zéro. Si c'est le cas, nous avons terminé, sinon nous calculons les solutions polynomiales modulo d'autres nombres premiers pertinents. Pour assurer la terminaison de ce procédé, nous devons montrer l'existence d'une borne sur la taille des coefficients des solutions polynomiales en caractéristique zéro.

Proposition 18. *Soit* $L=\sum_{i=0}^{n} a_i\,\partial^i \in \mathbb{Z}[x][\partial]$ *avec* $a_i=\sum_{j=-A}^{B} a_{i,j}\,x^{i+j}$ *où* A *et* B *sont définies par l'équation (4.1). Soit* N *une borne sur le degré d'éventuelles solutions*

polynomiales de L. Si $\mathcal{B}$ est la base échelonnée de solutions polynomiales (dans $\mathbb{Q}[x]$) de L, alors la taille du numérateur et du dénominateur de ses éléments est bornée par $(N+1)^{(N+1)/2}\, M^{(N+1)}$ où

$$M = \max_{i,k} |\alpha_i(k)|,$$

pour $-A \leq i \leq B$ et $0 \leq k \leq N + A + B$, avec

$$\alpha_i(k) = \sum_{j=0}^{n} a_{j,i}\, \frac{k!}{(k-j)!} \in \mathbb{Z}.$$

Démonstration. Comme nous l'avons vu dans la section 4.2.1, dès que nous avons une borne N sur le degré des solutions polynomiales de L, nous écrivons $y = \sum_{i=0}^{N} y_i\, x^i$ une éventuelle solution polynomiale et nous avons alors un système linéaire S pour les y_i. Ce système est de dimension $(N + A + B + 1) \times (N + 1)$. Notons r le rang de ce système (nous savons alors que la dimension de l'espace des solutions est égal à $N + 1 - r$). Pour résoudre ce système nous sommes donc amenés à résoudre un système (non-homogène) principal S_r de taille $r \times r$. Supposons que ce système soit formé des r premières lignes de S. Comme le déterminant Δ de S_r est non-nul, nous pouvons appliquer les formules de Cramer et écrire la solution générale de S_r sous la forme $(y_0, \ldots, y_{r-1})$ avec $y_i = \frac{\Delta_i}{\Delta}$ où Δ_i est le déterminant du système S_r où l'on a remplacé la $(i+1)$-ème colonne par le second membre (qui est une combinaison linéaire des inconnues non-principales $y_r, \ldots, y_N$). La taille du dénominateur des y_i est donc bornée par celle de $|\Delta|$. D'après la section 4.2.1, les coefficients le S_r sont des $\alpha_i(k)$ avec $-A \leq i \leq B$ et $0 \leq k \leq N + A + B$ donc à l'aide de l'inégalité d'Hadamard ([GG99, Theorem 16.6]), nous pouvons majorer $|\Delta|$ par $r^{r/2}\, \max_{i,k} |\alpha_i(k)|^r$. Chaque Δ_i s'écrit $\sum_{i=r}^{N} \delta_i\, y_i$ où les δ_i sont eux aussi majorés par $r^{r/2}\, \max_{i,k} |\alpha_i(k)|^r$ et les y_i pour $i = r, \ldots, N$ sont ici considérés comme des paramètres. Pour obtenir une base des $N + 1 - r$ solutions polynomiales, il nous suffit alors de prendre les solutions données par $(y_r, \ldots, y_N) = (1, 0, \ldots, 0), (0, 1, 0, \ldots, 0), \ldots$ dans ce qui précède. On obtient alors le résultat annoncé en majorant r par $N + 1$. □

Remarque 21. *Dans la démonstration précédente nous avons majoré « grossièrement » la taille des coefficients et il ne fait aucun doute que la borne que nous exhibons peut être améliorée. Par exemple, si l'on tient compte de la forme très particulière du système linéaire S et que l'on écrit soigneusement de proche en proche les solutions, on peut obtenir $2^{N-1}\, M^N$ (resp. M^N) comme borne sur la taille des numérateurs (resp. dénominateurs) des éléments de la base échelonnée de solutions polynomiales de degré inférieur ou égal à N de L. Il semble tout de même que, de toute manière, cette borne soit « intrinsèquement non optimale », dans le sens où on ne connait actuellement pas de borne* a priori *(comme la notre) qui soit atteinte sur un exemple.*
Pour obtenir des bornes sur la taille des coefficients des solutions polynomiales, nous pouvons aussi utiliser le lemme de Siegel (voir par exemple [BV83] ou [Ch01, 1.5]) : nous avons vu que calculer les solutions polynomiales revient à résoudre un système linéaire. Or le lemme de Siegel (classique) montre l'existence d'une solution non-nulle de « petite » taille pour un système linéaire sous-déterminé. Ceci est surtout intéressant si l'on cherche à borner la taille d'une seule solution.

Exemple 27. *Reprenons le cas de l'opérateur donné dans l'exemple 15. En cherchant ses solutions polynomiales nous avons écrit le système linéaire satisfait par leurs coefficients de sorte que la proposition 18 nous donne* $B = 5^{5/2}\,24^5 = 199065600\,\sqrt{5}$ *comme borne sur la taille des numérateurs et dénominateurs des coefficients de la solution polynomiale. On voit donc ici que cette borne est nettement supérieure à la taille réelle des coefficients de l'élément de la base échelonnée de solutions polynomiales de* L *qui est* $\{x^4 - 24\,x^3 + 144\,x^2 - 192\,x\}$.

4.6.2 Algorithme

Supposons que l'on dispose de l'ensemble $\mathcal{I}(L,\infty)$ défini dans la sous-section 4.1.2 (et donc d'une borne N sur le degré). Alors, on peut calculer récursivement une base échelonnée de solutions polynomiales dans $\mathbb{Q}[x]$ de $L \in \mathbb{Z}[x][\partial]$ comme suit :

Algorithme RecPolySols

Entrée : L, $[p_1,\ldots,p_{s-1},p_s]$, $[\mathcal{B}_{p_1},\ldots,\mathcal{B}_{p_{s-1}}]$.
Sortie : la base échelonnée $\mathcal{B}$ de solutions polynomiales non-nulles de L ou $\emptyset$.

1. Calculer $\mathcal{B}_{p_s}$.
2. Si $\mathcal{B}_{p_s} = \emptyset$, alors retourner $\emptyset$.
3. Sinon :
   3a. Si $\deg_x(\mathcal{B}_{p_s}) \not\subseteq \mathcal{I}(L,\infty)$, alors
   retourner RecPolySols(L,$[p_1,\ldots,p_{s-1},\mathrm{next}(p_s)]$,$[\mathcal{B}_{p_1},\ldots,\mathcal{B}_{p_{s-1}}]$).
   3b. Sinon :
   (i). Si $s>1$ et $\#(\mathcal{B}_{p_s}) > \#(\mathcal{B}_{p_{s-1}})$, alors
   retourner RecPolySols(L,$[p_1,\ldots,p_{s-1},\mathrm{next}(p_s)]$,$[\mathcal{B}_{p_1},\ldots,\mathcal{B}_{p_{s-1}}]$).
   (ii). Si $s=1$ ou $\#(\mathcal{B}_{p_s}) < \#(\mathcal{B}_{p_{s-1}})$, alors :
   - Appliquer l'algorithme CombPolySols aux éléments de $\mathcal{B}_{p_s}$,
   (on obtient ainsi une base $\mathcal{B}$ candidate)
   - Tester si les éléments y de $\mathcal{B}$ vérifient tous $L(y)=0$,
   - Si oui, alors retourner $\mathcal{B}$,
   - Sinon retourner RecPolySols(L,$[p_s,\mathrm{next}(p_s)]$,$[\mathcal{B}_{p_s}]$).
   (iii). Appliquer l'algorithme CombPolySols aux éléments de $\mathcal{B}_{p_1},\ldots,\mathcal{B}_{p_s}$
   (on obtient ainsi une base $\mathcal{B}$ candidate),
   (iv). Tester si les éléments y de $\mathcal{B}$ vérifient tous $L(y)=0$,
   (v). Si oui, alors retourner $\mathcal{B}$,
   (vi). Sinon, retourner RecPolySols(L,$[p_1,\ldots,p_s,\mathrm{next}(p_s)]$,$[\mathcal{B}_{p_1},\ldots,\mathcal{B}_{p_s}]$).

En entrée, $p_1,\ldots,p_{s-1}$ désignent des premiers pertinents modulo lesquels nous avons déjà calculé des bases échelonnées $\mathcal{B}_{p_1},\ldots,\mathcal{B}_{p_{s-1}}$ de solutions polynomiales et tels que les $\mathcal{B}_{p_i}$ ne suffisent pas à reconstruire une base échelonnée de solutions polynomiales en caractéristique zéro. Dans ce cas, nous avons besoin de plus de calculs modulaires et nous considérons donc un nouveau premier pertinent p_s. Initialement, nous ne connaissons

aucune base modulaire et nous appelons donc l'algorithme avec l'entrée $(L, [p_1], [])$ où p_1 est le plus petit nombre premier pertinent pour L.
Dans l'algorithme ci-dessus next(p_s) désigne le plus petit nombre premier pertinent strictement supérieur à p_s.
Dans l'étape 3a., $\deg_x(\mathcal{B}_{p_s})$ désigne l'ensemble des degrés des éléments de $\mathcal{B}_{p_s}$ de sorte que si $\deg_x(\mathcal{B}_{p_s}) \not\subseteq \mathcal{I}(L, \infty)$, alors nous ne considérons pas les calculs modulo p_s. En effet, le fait que $\deg_x(\mathcal{B}_{p_s}) \not\subseteq \mathcal{I}(L, \infty)$ signifie que p_s est un mauvais premier.

Dans l'algorithme RecPolySols tel qu'il est décrit ci-dessus, nous n'utilisons pas explicitement la borne exhibée dans la proposition 18 sur la taille des coefficients des solutions polynomiales de L. En effet, si nous voulions l'utiliser, alors il nous faudrait calculer des bases modulaires jusqu'à ce que le produit des premiers pour lesquels nous avons une base échelonnée soit supérieur à cette borne. Or, comme nous l'avons fait remarquer cette borne n'est pas optimale de sorte que ce procédé nous obligerait à considérer beaucoup plus de premiers que nécessaire.

Dans le procédé décrit ci-dessus, il se peut qu'à une certaine étape la reconstruction nous fournisse un certain nombre de solutions polynomiales de L mais pas la totalité c'est-à-dire pas une base. Dans ce cas avant de continuer, nous pouvons retirer les solutions déjà trouvées ce qui va réduire l'ordre de l'opérateur différentiel à considérer et pourrait donc accélérer les calculs suivants (à condition que cela n'augmente pas trop la taille des coefficiens de l' opérateur).

Pour résumer nous donnons ici le schéma général de l'algorithme modulaire que nous proposons ici :

Algorithme ModPolySols

Entrée : $L \in \mathbb{Z}[x][\partial]$,
Sortie : la base échelonnée de solutions polynomiales non-nulles de L ou $\emptyset$.

1. Calculer $\mathcal{I}(L, \infty)$. Si $\mathcal{I}(L, \infty) = \emptyset$, alors retourner $\emptyset$.
   Soit N son plus grand élément.
2. Appliquer le test rapide de non-existence de solutions polynomiales non-nulles (voir Section 4.4.1) pour $p = 2$, 3, 5. Si le test est positif, alors retourner $\emptyset$.
3. Appliquer l'algorithme RecPolySols.

Proposition 19. *Soit $L \in \mathbb{Z}[x][\partial]$, A et B définis par (4.1) et N (resp. b) une borne sur le degré (resp. la taille des coefficients - voir Proposition 18) d'éventuelles solutions polynomiales de L. Notons $h := \log(b)$ une borne sur la* hauteur *des coefficients des solutions cherchées. Alors, l'algorithme* ModPolySols *donné ci-dessus termine et retourne une base échelonnée de solutions polynomiales de L en $\mathcal{O}(h^3\, N\, n^2\, (A + B))$ opérations de mots.*

Démonstration. Pour montrer la correction de l'algorithme, il nous suffit de montrer la correction (terminaison) de l'algorithme RecPolySols. Tout d'abord, si L ne possède pas

de solution polynomiale non-nulle, alors d'après le théorème 4, nous ne pouvons trouver des solutions polynomiales de degré $\leq N$ modulo p que pour un nombre fini de p et nous détecterons donc nécessairement cette non-existence. Supposons maintenant que L possède des solutions polynomiales. Nous savons qu'il existe une infinité de nombres premiers pertinents (voir Proposition 17) de sorte que le nombre de premiers pertinents que nous pouvons considérer n'est pas limité. Or, pour p assez grand (supérieur au plus grand mauvais premier), toutes les bases modulaires $\mathcal{B}_{p_i}$ auront le même nombre d'éléments. Par conséquent, pour p assez grand, à chaque passage (récursif) dans l'algorithme RecPolySols, soit nous trouvons la base échelonnée de solutions polynomiales en caractéristique zéro, soit le produit des premiers pertinents considérés augmente. La taille des coefficients des solutions polynomiales cherchées est bornée par b : par conséquent, en considérant suffisament de p_i le produit P des p_i satisfait $\sqrt{P/2} > b$ et nous pouvons alors assurer qu'à ce moment là, la base échelonnée de solutions polynomiales que nous allons reconstruire à l'aide de notre algorithme CombPolySols sera la bonne (voir la sous-section 4.5.3).
Pour pouvoir reconstruire les solutions nous avons besoin de $h = \log(b)$ nombres premiers pertinents. Maintenant si on utilise la version modulaire de l'agorithme ABP, alors calculer les solutions modulaires coûte $\mathcal{O}(h\,N\,n^2\,(A+B))$ opérations de mots : en effet on peut majorer « grossièrement » la taille des entiers sur lesquels on calcule par h. La reconstruction coûte elle à chaque fois au plus $h^2\,N\,n$ opérations de mots. Enfin, pour chaque polynôme y reconstruit, la vérification $L(y) = 0$ peut se faire en un coût équivalent à celui du calcul des solutions polynomiales : en effet calculer la solution polynomiale générale de L peut se faire en déroulant la récurrence linéaire associée à L et vérifier qu'un polynôme y satisfait $L(y) = 0$ peut se faire en vérifiant que ses coefficients satisfont cette même récurrence linéaire ce qui du point de vue de la complexité est équivalent (voir aussi [Bos03, Ch10, p. 294]). On obtient donc le résultat de complexité annoncé. □

Remarque 22. *Une manière d'améliorer notre méthode consiste à ne pas reconstuire les solutions à chaque fois que l'on ajoute une base modulaire mais seulement lorsque l'on dispose de plusieurs nouvelles bases modulaires. C'est ainsi que procèdent les auteurs dans [HM02] pour développer un algorithme modulaire calculant le* pgcd *de polynômes à coefficients dans des corps de nombres $\mathbb{Q}(\alpha_1, \ldots, \alpha_r)$ avec les α_i algébriques sur $\mathbb{Q}$. L'intérêt est que cela diminue le coût de la reconstruction. Dans cette reconstruction le problème vient de la reconstruction rationnelle et non de l'algorithme des restes chinois qui lui peut se faire au fur et à mesure pour un coût total en $\mathcal{O}(h^2)$. Comme cela est remarqué dans [HM02], en utilisant cette amélioration, on peut gagner un facteur h dans la complexité de notre algorithme donnée par la proposition 19.*

4.7 Conclusion et perspectives

Si l'on compare la complexité de notre algorithme modulaire (Proposition 19) à celles des algorithmes ABP et Barkatou (Propositions 14 et 15), on s'aperçoit que :

- les dépendances en N, n et $A + B$ sont, dans tous les cas, linéaires,
- la dépendance en h est linéaire pour ABP et Barkatou mais, à cause de la reconstruction, elle est cubique pour notre algorithme modulaire (si l'on utilise l'amélioration de la remarque 22, alors la dépendance reste quadratique).

Par conséquent, notre analyse de complexité montre que notre algorithme modulaire perd « en général » d'un facteur h par rapport à ABP et Barkatou.

Pour pouvoir observer cette différence en pratique, nous avons comparé des implantations de notre algorithme et de ABP et Barkatou. Nous avons procédé comme suit : nous fixons des bornes sur le nombre de solutions polynomiales linéairement indépendantes, la taille de leur degré et de leurs coefficients et nous construisons au hasard des polynomes satisfaisant ces choix. Ensuite nous construisons l'opérateur différentiel admettant ces polynômes comme base de solutions polynomiales et nous observons les temps de calculs des différents algorithmes. La conclusion est que, conformément à ce que l'on peut attendre, notre algorithme est en général moins rapide que ABP ou Barkatou. La différence est surtout significative lorsque les coefficients de l'opérateur sont petits par rapport à ceux des éléments d'une base échelonnée de solutions polynomiales (note : ceci n'est pas une surprise). Typiquement, pour l'opérateur donné dans l'exemple 15, dès que l'on fait croître un peu le paramètre m, notre algorithme met beaucoup plus de temps que les deux autres pour calculer la base échelonnée de solutions polynomiales.

Cependant lorsque les coefficients de la sortie sont plus petits que ceux de l'opérateur ou que ceux intervenants dans les calculs intermédiaires, alors notre algorithme devient compétitif :

Exemple 28. *Si l'on considère l'opérateur d'ordre* 2 *possédant comme solutions* $1+x+\cdots+x^d$ *(avec* $d \in \mathbb{N}^*$*) et* $\exp(x)$*, alors notre algorithme est plus rapide qu'*ABP *ou* Barkatou*. À titre d'indication, voici les temps (en secondes) obtenus pour certaines valeurs de d par les différents algorithmes considérés.* ABP *et* Barkatou *sont des versions des algorithmes du même nom implantées respectivement par M. Bronstein dans Bernina*[3] *et M. A. Barkatou dans Maple.* ModPolySols *est une implantation par M. Bronstein en Bernina d'un algorithme très proche de notre algorithme modulaire. Les calculs ont été faits dans Maple 9 (Bernina a une interface avec Maple) sur une machine possédant les caractéristiques suivantes : processeur Pentium M* 1.6 *GHz, Intel* 855*GME chipset,* 760 *de RAM.*

[3]http ://www-sop.inria.fr/cafe/Manuel.Bronstein/sumit/bernina.html

d	*temps(*ABP*)*	*temps(*Barkatou*)*	*temps(*ModPolySols*)*
100	0.71	0.159	0.24
200	4.08	0.519	0.49
300	12.64	1.176	0.95
400	30.63	2.063	1.57
500	62.87	3.428	2.33
800	310.43	9.346	6.41
1000	682.16	14.729	8.12

Ces temps peuvent s'interpréter de la manière suivante : L'algorithme ABP *va calculer le début du développement d'une base de solutions sous forme de séries en* 0*; il calculera donc le début du développement de l'exponentielle en* 0 *bien que ceci soit inutile. Quant à la différence entre les temps obtenus pour notre algorithme et celui de* Barkatou*, elle s'explique par le fait que dans notre algorithme les calculs vont se faire modulo p au lieu de se faire dans* $\mathbb{Q}$*. Sur cet exemple particulier, le fait que* Barkatou *soit plus rapide que* ABP *vient uniquement du fait que* Barkatou *se place à l'infini et ne calcule donc pas le développement de l'exponentielle. Il est cependant possible de construire un exemple où le cas contraire se produit,* i. e.*, il y a deux séries à développer à l'infini et une seule en zéro de sorte que* ABP *sera alors plus rapide que* Barkatou*.*

Ces observations nous permettent d'exhiber des familles naturelles d'opérateurs pour lesquels notre algorithme sera meilleur que les algorithmes existants : typiquement, les constructions (puissances symétriques, extérieures,...) sur les opérateurs différentiels sont des objets éliminants, *i. e.*, venant d'une élimination type vecteur cyclique, dont les coefficients sont « artificiellement gros » par rapport à la taille de leurs solutions. Le calcul des solutions polynomiales de telles constructions est très utile en théorie de Galois différentielle où l'on discrimine les groupes en fonction du fait que certaines constructions possèdent ou non des solutions rationnelles.

Exemple 29. *Considérons l'opérateur d'ordre* 3

$$L := Dx^3 + \frac{(-2069\,x + 744 + 800\,x^2)}{900\,x^2\,(x-1)^2}\,Dx - \frac{-18985\,x^2 + 13987\,x - 4352 + 4800\,x^3}{5400\,x^3\,(x-1)^3},$$

et cherchons à déterminer son groupe de Galois différentiel (ses solutions liouvilliennes) en utilisant l'algorithme donné dans [HRUW98, 4.1]. Après avoir montré que L est irréductible et que son groupe de Galois différentiel ne peut pas être imprimitif, on s'intéresse au calcul d'invariants. En considérant la 2*-ème puissance symétrique* $L^{\circledS_2}$ *de L, on montre l'existence d'un invariant de degré* 2 *et on est donc conduit à chercher des invariants de degré* 6*. Pour ce faire nous devons chercher les solutions polynomiales de la* 6*-ème puissance symétrique* $L^{\circledS_6}$ *de L :* $L^{\circledS_6}$ *est un opérateur d'ordre* 28 *ayant de « très gros » coefficients ; pour donner une idée de la taille de ces coefficients, le numérateur du coefficient de* ∂^{27} *(c'est-à-dire du plus petit coefficient) est un polynôme dense de degré* 108 *dont le début du développement est :*

$$489904476132347381519612264668922728869378693421973475247664641376462093922776199623605822173620185739730311$$
$$351885382182886355349883802754330427941852143446041475840477361110377849807871955600925625000000 x^{108}+$$
$$38844491602707186844562153667686100686551653230442805695437092503288295664557796631141791223686552850640233714629$$
$$28580095874444172886202735457578535161827995581068583739038425413164503760884232805793058593750 0 x^{107}+$$
$$58577146702965135563884176173233708654537513692189746657327129904932194887334259886737671769408095470580 4$$
$$74319860927727996526399792223838370982027670170462877836062745655934631381554010765073082857467765474687 50 x^{106} + \cdots .$$

Sa solution polynomiale générale est :

$$y = c_1\, x^6 - \frac{18667}{25}\, c_1\, x^5 + (-\frac{6040112178}{171475}\, c_1 + \frac{8741816}{6859}\, c_2)\, x^4$$
$$+ (-\frac{112559383}{9025}\, c_1 + \frac{127308}{361}\, c_2)\, x^3 + (\frac{618}{19}\, c_2 - \frac{151436}{475}\, c_1)\, x^2 + c_2\, x + \frac{81}{25}\, c_1.$$

Le fait que la taille des coefficients de cette solution polynomiale générale soit « très petite » comparée à celle des coefficients de $L^{\circledS_6}$ fait que notre algorithme modulaire sera plus efficace que les algorithmes existants ; si on compare les temps de calcul des trois algorithmes déjà considérés dans l'exemple précédent, alors on obtient les temps suivants (en secondes) :

*temps(*ABP*)*	*temps(*Barkatou*)*	*temps(*ModPolySols*)*
13.3	13.8	9.3

L'espace des solutions polynomiales de $L^{\circledS_6}$ est donc de dimension 2 *ce qui signifie que L possède deux invariants de degré* 6 *; on peut donc conclure que le groupe de Galois de L est isomorphe à $\mathrm{A}_5^{\mathrm{SL}_3}$ et en déduire les solutions liouvilliennes de L.*

La conclusion de cette étude modulaire du calcul des solutions polynomiales d'opérateurs différentiels linéaires est qu'actuellement, un algorithme modulaire n'est pas universellement meilleur que les algorithmes existants pour le problème du calcul d'une base de solutions polynomiales. Par contre, notre approche modulaire est intéressante pour :

- détecter la non-existence de solution polynomiale non-nulle ou même deviner la dimension de l'espace des solutions polynomiales (problème utile en théorie de Galois différentielle),
- calculer les solutions polynomiales d'opérateurs différentiels ayant de « gros » coefficients mais de « petites » solutions polynomiales. De tels opérateurs apparaissent naturellement dans les contructions en théorie de Galois différentielle.

Dans [Bos03], Bostan montre que calculer le N-ème terme d'une récurrence linéaire à coefficients polynomiaux peut se faire en $\mathcal{O}(\sqrt{N})$ opérations dans le corps de base. Un thème de nos recherches actuelles est d'utiliser ceci pour :

- tester de manière encore plus efficace la non-existence de solution polynomiale non-nulle,

– calculer probabilistiquement la dimension de l'espace des solutions polynomiales,
– calculer une base de solutions polynomiales.

Appendice

Systèmes différentiels

Nous donnons ici les procédures servant à factoriser un système différentiel en caractéristique p. Ces procédures sont écrites dans le logiciel de calcul formel Maple ; elles sont disponibles à
`http://www.unilim.fr/pages_perso/thomas.cluzeau/CluzeauModularFactorization`
avec quelques explications sur les méthodes utilisées. Les packages nécessaires sont *DEtools* et *linalg*.

Les systèmes différentiels que nous considérons sont supposés à coefficients dans $\mathbb{F}_p(z)$.

p-courbure

La procédure suivante calcule la p-courbure d'un système $Y' = AY$ avec $A \in \mathbb{M}_n(\mathbb{F}_p(z))$ en suivant la méthode sans dénominateurs détaillée dans la sous-section 1.3.1. La variable z est donnée en entrée : ceci nous permet d'utiliser aussi cette procédure pour calculer les p-courbures partielles dans le cas des systèmes d'équations aux dérivées partielles. Les variables _Lie_Pol et _d sont globales ; ceci nous permet de les ré-utiliser si l'on souhaite cacluler une matrice fondamentale de solutions rationnelles avec la « formule de Katz » (voir Sous-section 1.3.5).

```
LiePol := proc (A, n, p, z)
local i, j, B, S, De, d1 ;
global _Lie_Pol, _d ;
    De := map(denom, A) ;
    _d := lcm(seq( seq(De_{i,j}, j=1 .. n), i=1 .. n)) ;
    d1 := diff(_d, z) ;
    B := evalm(_d * A) ;
    _Lie_Pol_0 := Matrix(n, n, shape=identity) ;
    _Lie_Pol_1 := - B ;
    for i from 2 to p do
    _Lie_Pol_i := map(X -> modp(Normal(X), p), evalm (
        evalm(_d * diff_mat(_Lie_Pol_{i-1}, p)) - '&*'(B, _Lie_Pol_{i-1})
         - evalm((i - 1) * d1 * _Lie_Pol_{i-1})))
    end do;
    RETURN(map(X -> modp(Normal(X), p), evalm(_Lie_Pol_p/_d^p)))
end proc
```

Polynôme caractéristique

La procédure suivante calcule le polynôme caractéristique d'une matrice $M \in \mathbb{M}_n(\mathbb{F}_p(z))$. Comme pour nos algorithmes les matrices dont nous devons calculer le polynôme caractéristique sont à coefficients polynomiaux, nous n'utilisons pas la procédure *charpoly* de Maple, peu efficace lorsque les matrices sont à coefficients polynomiaux. Sur une idée de M. van Hoeij, nous utilisons plutôt une méthode d'interpolation ; pour disposer d'assez de points pour pouvoir interpoler, p doit être choisi strictement plus grand que n.

```
Char_Pol_modp := proc (M, λ , z, p)
local n, Id, Mat, i, S, X , Mat_pol ;
    n := rowdim(M) ;
    Id := Matrix(n, n, shape=identity) ;
    Mat := evalm(λ * Id − M) ;
    Mat_pol := map(X → simplify(X) mod p, concat( seq(
       evalm(simplify(col( Mat, i)* lcm(seq(denom( col(Mat, i)_j), j=1 .. n)))), i=1
.. n))) ;
    X := [seq(i, i=0 .. n)] ;
    S := [seq(Det(evalm( map(v → subs(λ =X_i, v), Mat_pol))) mod p, i=1 .. n +
1)] ;
    RETURN(
       collect(Rem(Primpart( Interp(X, S, λ ) mod p, λ) mod p, z^p - c, z) mod p,
λ ))
end proc
```

LBTM et UBTM

Ces deux procédures construisent des matrices de Toeplitz par blocs (inférieures ou supérieures) ; ceci nous sera utile pour construire la matrice de l'opérateur Δ_A appelée N dans la sous-section 1.2.1.

```
LBTM := proc (L, r, n, p)
local N, j, T ;
    if 0 < r then
       N := matrix(n, n, 0) ;
       T := augment(stackmatrix( op(L)),
          seq(stackmatrix(N $ j, op(L)_{1..r−j}), j=1 .. r - 1))
    end if
end proc
```

```
UBTM := proc (L, r, n)
local N, j, T ;
    if 0 < r then
       N := matrix(n, n, 0) ;
       T :=
          stackmatrix(augment(op (L)), seq(augment(N $ j, op(L)_{1..r−j}), j=1 .. r -
```

```
1))
    end if
end proc
```

Décomposition d'une fraction rationnelle sur la base $(1, z, \ldots, z^{p-1})$

Cette procédure décompose une fraction rationnelle appartenant à $\mathbb{F}_p(z)$ dans la base $(1, z, \ldots, z^{p-1})$ de $\mathbb{F}_p(z)$ sur $\mathbb{F}_p(z^p)$.

```
pdecomposition := proc (f, p)
local PP, i, f1, Q, P1, S1, R, Q1, j;
    f := Normal(F) mod p ;
    PP := numer(f) ;
    Q := denom(f) ;
    P1 := Rem(PP, z^p - c, z) mod p ;
    Q1 := Rem(Q, z^p - c, z) mod p ;
    Gcdex(Q1, z^p - c, z, 'S1', 't') mod p ;
    R := Rem(P1 * S1, z^p - c, z) mod p ;
    for j from 0 to p - 1 do f1_{j+1} := coeff(R , z, j) mod p end do ;
    RETURN(seq(f1_i, i=1 .. p))
end proc
```

Décomposition d'une matrice sur la base $(1, z, \ldots, z^{p-1})$

```
pdecomposition_mat := proc (A, n, p)
local CC, i, Mat;
    CC := convert(map( pdecomposition, A, p, α ), array) ;
    for i to p do Mat_i := concat(seq(col( CC, i + j * p), j=0 .. n - 1)) end do ;
    RETURN(seq(evalm( Mat_i), i=1 .. p))
end proc
```

Dérivée terme à terme d'une matrice

```
diff_mat := proc(A, p) RETURN(map(modp, map(diff, evalm(A), z), p)) end proc
```

Matrice de l'opérateur Δ_A dans la base canonique de $\mathbb{F}_p(z)^n$ sur $\mathbb{F}_p(z^p)^n$

```
matrice_op := proc (A, n, p)
local i, j, k, L, l, d, E, M, w, v, v1, v2, R1, q, r;
    M := matrix(p*n, p*n) ;
    v2 := vector(n) ;
    for i to p*n do
        d := iquo(i, p, 'r') ;
        if r ≠ 0 then q := d + 1 else q := d end if ;
        v := vector(n, 0) ;
        r := (i - 1) mod p ;
        v_q := z^r ;
        v2 := evalm(map( proc (f) diff(f, z) end proc , v ) - multiply(A, v)) ;
        L_i := convert(map( pdecomposition, v2, p, α ), array )
    end do;
    M := concat(seq(L_i, i=1 .. p*n)) ;
    RETURN(M)
end proc
```

Matrice de l'opérateur Δ_A dans une base pertinente

Cette procédure calcule la matrice de l'opérateur Δ_A appelée N dans la sous-section 1.2.1. Cette matrice étant très struturée, son noyau se calcule plus facilement que celui de la matrice de l'opérateur dans la base canonique que l'on obtient avec la procédure précédente.

```
mat_op_bp := proc (A, n, p)
local SS, TildeN, Z, Id, L, Mat_Der, N, i;
    SS := [pdecomposition_mat(A, n, p)] ;
    TildeN := evalm(LBTM([ seq(SS_i, i=1 .. p)], p, n)
        + UBTM([diag(0 $n), seq(c* SS_{p-i}, i =0 .. p - 2)], p, n)) ;
    Z := Matrix(n, n, shape= zero) ;
    Id := Matrix(n, n, shape=identity) ;
    for i to p - 1 do
L_i := concat(seq(Z, j=1 .. i), - i * Id, seq(Z, j=i + 2 .. p))
    end do;
    L_p := concat(seq(Z, i=1 .. p)) ;
    Mat_Der := map(modp, stackmatrix(seq(L_i, i=1 .. p)), p) ;
    N := map(modp, evalm (TildeN + Mat_Der), p) ;
    RETURN(N)
end proc
```

Solutions Rationnelles 1

Cette procédure calcule les solutions rationnelles de $Y' = A\,Y$ en calculant le noyau de la matrice de l'opérateur Δ_A dans la base canonique.

```
rat_sol1 := proc (A, n, p)
local B, N1, N, N2, i, j, l, k, De, d;
    B := evalm(matrice_op(A, n, p)) ;
    De := map(denom, B) ;
    d := lcm(seq(seq( De_{i,j}, j=1 .. p*n), i=1 .. p*n)) ;
    N1 := Nullspace(evalm( d*B)) mod p ;
    N2 := matrix(nops( N1), n) ;
    k := 0 ;
    for i to nops( N1) do
       for j to n do N2_{i,j} := add(N1_{il+k+1} *z^l, l=0 .. p - 1) ;k := k + p end do;
       N_i := [seq(N2_{i,j}, j=1 .. n)] ;
       k := 0
    end do;
    RETURN(seq(N_i, i=1 .. nops(N1)))
end proc
```

Solutions Rationnelles 2

Cette procédure calcule les solutions rationnelles de $Y' = A\,Y$ en calculant le noyau de la matrice de l'opérateur Δ_A dans une base pertinente.

```
rat_sol2 := proc (A, n, p)
local i, j, NN, N1, s, y, De, d;
    NN := evalm(mat_op_bp (A, n, p)) ;
    De := map(denom, NN) ;
    d := lcm(seq(seq( De_{i,j}, j=1 .. p*n), i=1 .. p*n)) ;
    N1 := Nullspace(evalm( d*NN)) mod p ;
    for i to nops( N1) do
       for j from 0 to p - 1 do y_{i,j} := vector(n, [ seq(N1_{ik+n*j}, k=1 .. n)]) end
do
    end do;
    for i to nops( N1) do s_i := add(y_{i,j} *z^j, j=0 .. p - 1) end do ;
    RETURN(seq(evalm(s_i), i=1 .. nops(N1)))
end proc
```

Factorisation Absolue

Dans le cas où nous voulons factoriser notre système sur $\overline{\mathbb{F}}_p(z)$, nous avons besoin de calculer des factorisations absolues du polynôme caractéristique de la p-courbure : nous le faisons à l'aide de cette procédure.

```
AbsFact := proc (f, p, mult)
local test, n, m, g, S, minim, R, i;
```

```
    test := 0 ;
    n := degree(f, T) ;
    m := degree(f, z) ;
    g := gcd(n, m) ;
    if g=1 then RETURN ([f, mult])
    else
       while test=0 do
          minim := randpoly(x, degree=g, coeffs=rand(0 .. p - 1)) ;
          if Irreduc(minim) mod p=true and degree( minim, x)=g and lcoeff(
minim)=1
          then test := 1
          else test := 0
          end if
       end do;
       print(minim, f) ;
       S := Factors(f, RootOf( minim)) mod p
    end if;
    for i to nops( op(2, S)) do
       R_i := [op(1, op(i, op(2, S))), mult* op(2, op(i, op(2, S))]
    end do;
    RETURN(seq(R_i, i=1 .. nops(op(2, S))))
end proc
```

p-courbure scalaire

Cette procédure diagonalise un système $Y' = AY$ n'admettant pas 0 comme point singulier et ayant une p-courbure scalaire ; pour des détails voir l'algorithme Scal$p$Curv dans la sous-section 1.3.5.

```
ScalpCurv := proc (A, n, p)
local i, μ , Id, B, Inter;
global _PP, _IPP, _M ;
    μ := simplify( - trace(A)/n) mod p ;
    Id := Matrix(n, n, shape=identity) ;
    B := evalm(A + μ * Id) ;
    LiePol(B, n, p) ;
    _PP := map(X → Normal(X) mod p,
       evalm(add((−z)^i * _Lie_Pol_i/(i ! * _d^i), i = 0..p − 1))) ;
    _IPP := Inverse(_PP) mod p ;
    Inter := map(X→ modp(Normal(X), p), evalm( '&*'('&*'(_IPP, A), _PP))) ;
    _M :=
       map(X→ modp( Normal(X), p), evalm(Inter - '&*' (_IPP, diff_mat(_PP,
p)))) ;
    RETURN(_M)
end proc
```

Décomposition isotypique

Cette procédure calcule une décomposition isotypique d'un système $Y' = AY$ où $A \in \mathbb{M}(\mathbb{F}_p(z))$; pour des détails voir l'algorithme IsoDec dans la sous-section 1.5.1. Le 4-ème argument en entrée est une option valant 0 ou 1 ; si cette option vaut 0, on factorise sur $\mathbb{F}_p(z)$ et si elle vaut 1 on factorise sur $\overline{\mathbb{F}}_p(z)$.

```
IsoDec := proc (A, Pcurv, p, opt)
local F, i, G, N, inter, j, l, GG, GGG ;
global _Nbre1, _Nbre2, _PP, _MMM, _G1, _IPP ;
    F := op(2, Factors( Char_Pol_modp(Pcurv, T, z, p)) mod p ) ;
    GGG := select(X→ has(X, T), F) ;
    for i to nops( GGG) do
       if 1 < degree(GGG_i1, T) and opt=1 then
          G_i := AbsFact(GGG_i1, p, GGG_i2)
       else G_i := GGG_i
       end if
    end do ;
    _G1 := [seq(G_i, i=1 .. nops(GGG))] ;
    _Nbre1 := nops(_G1) ;
    if _Nbre1=1 then
       _MMM := A ;
       _PP := Matrix(rowdim( A), shape=identity) ;
       _IPP := _PP ;
       _Nbre2_1 := rowdim(A)
    else
       for i to _Nbre1 do
          N_i :=
             Nullspace(evalm(subs(T =Pcurv, subs(c=z^p, _G1_i1^-G1_i2)))) mod p ;
          _Nbre2_i := nops(N_i)
       end do ;
       _PP :=
          concat(seq(seq( convert(N_ij, matrix), j=1 .. nops(N_i)), i=1 .. _Nbre1)) ;
       _IPP := Inverse(_PP) mod p ;
       inter := map(X→ modp(Normal(X), p), evalm( '&*'('&*'(_IPP, A), _PP))) ;
       _MMM :=
          map(X→ modp( Normal(X), p), evalm(inter - '&*' (_IPP, diff_mat(_PP,
p))))
    end if ;
    RETURN(_MMM)
end proc
```

Décomposition scalaire

Cette procédure calcule une décomposition isotypique de $Y' = AY$ où $A \in \mathbb{M}(\mathbb{F}_p(z))$ et diagonalise les blocs restants ayant une p-courbure scalaire.

```
ScalDec := proc (A, n, p, opt)
local
k, i, Pcurv, Id, Z, Id1, c , so, so1, c1, Pcurvbis, Pcurv1, B, A1, Z1, G2, multip,
m ;
    Id := Matrix(n, n, shape=identity) ;
    Z := Matrix(n, n, shape= zero) ;
    Pcurv := LiePol(A, n, p) ;
    if IsMatrixShape( convert(map(X→ modp(Normal( X), p), Pcurv), Matrix),
scalar )
    then RETURN((Normal( - trace(A)/n) mod p) *Id)
    else
        evalm(IsoDec(A, Pcurv , p, opt)) ;
        if _Nbre1=1 and op(2, op(1, _G1))=1 then print(irreducible) ;RETURN(A)
        else
            Pcurvbis :=
              map(X→ modp( Normal(X), p), evalm(`&*`(`&*`( _IPP, Pcurv), _PP))) ;
            k := 1 ;
            for i to _Nbre1 do
                A1_i := convert( _MMM, Matrix)_{k..k+_Nbre2_i-1, k..k+_Nbre2_i-1} ;
                Pcurv1_i := map(X → modp(Normal(X), p),
                    convert(Pcurvbis, Matrix)_{k..k+_Nbre2_i-1, k..k+_Nbre2_i-1}) ;
                k := k + _Nbre2_i
            end do;
            for i to _Nbre1 do
                if _Nbre2_i=1 then B_i := evalm(A1_i)
                else
                    Id1_i := Matrix( rowdim(Pcurv1_i), rowdim(Pcurv1 _i), shape=identity) ;
                    if IsMatrixShape( convert(map(X→ modp(Normal( X), p), Pcurv1_i),
Matrix),
                     scalar)then B_i := evalm((Normal( - trace(A1_i)/_Nbre2_i) mod p) *
Id1_i)
                    else B_i := evalm(A1_i)
                    end if
                end if
            end do
        end if;
        RETURN(
          evalm(diag(seq( map(X→ modp(Normal(X), p) , evalm(B_i)), i=1 .. _Nbre1))))
    end if
end proc
```

Eigenring

Cette procédure calcule l'eigenring d'un système différentiel $Y' = AY$. Le package *tensor* est nécessaire.

```
eigmodp := proc (A, n, p)
local i, j, k, l, TA, TRANSPA, ID, TENS, r, MAT, N, MAT1, RATSOL,
Outp, Ligne ;
```

```
    r := 1 ;
    TA := create([1, 1], convert(A, array)) ;
    TRANSPA := create([1, 1], convert(transpose(A), array)) ;
    ID := create([1, 1], convert(Matrix(1 .. n, 1 .. n, shape=identity), array)) ;
    TENS := lin_com(prod( TA, ID), -1, prod(ID, TRANSPA)) ;
    if op(2, op(1, op(2, op(1, TENS))))=[1, 1, 1 , 1] then
       MAT := op(2, op(2, op(2, op(1, TENS))))
    else MAT := op(2, op(1, op(2, op(1, TENS) )))
    end if ;
    N := array(1 .. n^2) ;
    for i to n do for j to n do
          N_r := [seq(seq( MAT_{i,k,j,l}, l=1 .. n), k=1 .. n)] ; r := r + 1
       end do
    end do ;
    MAT1 :=
       map(X→ modp( Normal(X), p), evalm(stackmatrix( seq(N_i, i=1 .. n^2)))) ;
    RATSOL := [rat_sol2( MAT1, n^2, p)] ;
    Outp := array(1 .. nops (RATSOL)) ;
    for i to nops( RATSOL) do
       k := 1 ;
       for j to n do  Ligne_j := [seq(RATSOL_{ij} , j=k .. n + k - 1)] ; k := k + n end
do ;
       Outp_i := stackmatrix( seq(Ligne_j, j=1 .. n))
    end do ;
    RETURN(
       seq(map(X→ modp(Normal(X), p), evalm(Outp_i)), i=1 .. nops(RATSOL)))
end proc
```

Systèmes d'ÉDP D-finis

La procédure suivante calcule une décomposition isotypique simultanée d'une famille de matrices qui commutent entre elles deux à deux au sens de la définition 23 ; pour des détails, voir la procédure IsoDecSim dans la sous-section 1.8.4.

```
IsoDec_Simult := proc (S, n)
local s, i, F, nF, v, m, vv, PP, IPP, j, k, B, P1 , C, A1, PP1 ;
    if S={} then  RETURN(diag(seq(1, i=1 .. n))) end if  ;
    s := nops(S) ;
    if IsMatrixShape( convert(S_1, Matrix), scalar) then
       RETURN(IsoDec_Simult(S minus {S_1}, n))
    else
       F := select(Z→ has(Z , X), (Factors(numer(charpoly(S_1, X))) mod p)_2) ;
       nF := nops(F) ;
       if nF=1 and F_{nF2}=1 then RETURN( diag(seq(1, i=1 .. n)))
       else
          if nF=1 then  RETURN(IsoDec_Simult(S minus {S_1}, n)) end if  ;
          for i to nF do
             v_i := Nullspace(evalm( subs(X=S_1,  F_{i1}^{F_{i2}}))) mod p ;
             m_i := nops(v_i) ;
```

```
            vv_i := concat(seq (v_ij, j=1 .. m_i))
          end do;
          PP := concat(seq(vv_i, i=1 .. nF)) ;
          IPP := Inverse(PP) mod p ;
          for i from 2 to s do
            B_i := map(X→ Normal(X) mod p, evalm('&*'( '&*'(IPP, S_i), PP)))
          end do;
          m_0 := 0 ;
          for k to nF do for j from 2 to s do C_j,k :=
              submatrix(B_j, m_k-1 + 1 .. m_k + m_k-1, m_k-1 + 1 .. m_k + m_k-1 )
            end do
          end do;
          for k to nF do P1_k := IsoDec_Simult({ seq(C_j,k, j=2 .. s)}, m_k) end
do ;
          PP1 := diag(seq( evalm(P1_k), k=1 .. nF)) ;
          RETURN(evalm('&*'( PP, PP1)))
        end if
      end if;
      RETURN(evalm('&*'( PP, PP1)))
end proc
```

Bibliographie

[Abh90] S. S. Abhyankar. Algebraic geometry for scientists and engineers. Mathematical Surveys and monographs, number **35**. Published by the A.M.S., 1990.

[Abr95] S. A. Abramov. Rational solutions of linear difference and q-difference equations with polynomial coefficients. In *(Russian) Programmirovanie*, **6** : 3-11, 1995 ; translation in *Program. Comput. Software 21*, **6** : 27-278, 1995.

[AB98] S. A. Abramov and M. A. Barkatou. Rational solutions of first order linear difference systems. In *Proceedings of ISSAC'98*, ACM Press, 124-131, 1998.

[ABP95] S. A. Abramov, M. Bronstein and M. Petkovšek. On polynomial solutions of linear operator equations. In *Proceedings of ISSAC'95*, ACM, New York, 290-296, 1995.

[AH03] S. A. Abramov and M. van Hoeij. Set of poles of solution of linear difference equations with polynomial coefficients. In *Computational Mathematics and Mathematical Physics*, **43**(1) : 57-62, 2003.

[AK91] S. A. Abramov and K. Yu. Kvansenko. Fast algorithm to search for the rational solutions of linear differential equations with polynomial coefficients. In *Proceedings of ISSAC'91*, ACM Press, 267-270, 1991.

[APP98] S. A. Abramov, P. Paule and M. Petkovšek. q-Hypergeometric solutions of q-difference equations. In *Discrete Math.*, **180** : 3-22, 1998.

[APR00] S. Abramov, M. Petkovšek and A. Ryabenko. Special formal series solutions of linear operator equations. *Discrete mathematics*, **210** : 3-25, 2000.

[Ad28] C. R. Adams. On the irregular cases of the linear ordinary difference equations. In *Trans. Amer. Math. Soc.*, **30** : 507-541, 1928.

[Ba93] M. A. Barkatou. An algorithm for computing the companion block diagonal form for a system of linear differential equations. In the journal *Appl. Alg. in Eng. Comm. and Comput.*, **4** : 185-195, 1993.

[Ba97] M. A. Barkatou. A fast algorithm to compute the rational solutions of systems of linear differential equations. *Rapport de Recherche*, **973-M**, IMAG-Grenoble, 1997.

[Ba99] M. A. Barkatou. On rational solutions of systems of linear differential equations. In *Journal of Symbolic Computation*, **28** : 547-567, 1999.

[Ba01] M. A. Barkatou. On the reduction of matrix pseudo-linear equations. Technical Report RR 1040, Rapport de Recherche de l'institut IMAG, 2001.

[BD94] M. A. Barkatou and A. Duval. Sur les séries formelles solutions d'équations aux différences polynômiales. In *Ann. Inst. Fourier*(Grenoble), **44**(2) : 495-524, 1994.

[BP98] M. A. Barkatou and E. Pflügel. On the equivalence problem of linear differential systems and its application for factoring completely reducible systems. In *Proceedings of ISSAC'98*, ACM Press, 268-275, 1998.

[Be1894] E. Beke. Die irreduzibilität der homogenen linearen differentialgleichungen. In *Math. Ann.*, **45** : 278-294, 1894.

[Bi30] G. D. Birkhoff. Formal theory of irregular linear difference equations. In *Acta Math.*, **54** : 205-246, 1930.

[BCGPR03] Y. A. Blinkov, C. F. Cid, V. T. Gerdt, W. Plesken and D. Robertz. The Maple package "Janet" : II. Polynomial Systems. In *CASC*, 2003.

[BV83] E. Bombieri and J. Vaaler. On Siegel's lemma. In *Invent. Math.*, **73(1)** : 11-32, 1983.

[Bom01] R. Bomboy. Réductibilité et résolubilité des équations aux différences finies. Thèse de l'Université de Nice-Sophia Antipolis, 2001.

[Bos03] A. Bostan. Algorithmique efficace pour des opérations de base en calcul formel. Thèse de l'École polytechnique, 2003.

[BLS03] A. Bostan, G. Lecerf and É. Schost. Tellegen's principle into practice. In *Proceeding of ISSAC'03*, ACM Press, 37-44, 2003.

[Bou00] D. Boucher. Sur les équations différentielles linéaires paramétrées, une application aux systèmes Hamiltoniens. Thèse de l'Université de Limoges, 2000.

[Br92] M. Bronstein. On solutions of linear ordinary differential equations in their coefficient field. In *Journal of Symbolic Computation*, **13** : 413-440, 1992.

[Br94] M. Bronstein. An improved algorithm for factoring linear ordinary differential operators. In *Proceedings of ISSAC'94*, ACM Press, 336-340, 1994.

[Ca94] J. Cai. Computing Jordan normal forms exactly for commuting matrices in polynomial time. In *International Journal of Foundations of Computer Sciences*, **5(3)** : 293-302, 1994.

[CO68] L. Chambadal and J. L. Ovaert. Algèbre linéaire et algèbre tensorielle. Dunot Université, Paris, 1968.

[Ch01] A. Chambert-Loir. Théorèmes d'algébricité en géométrie diophantienne. *Séminaire Bourbaki*, exposé No. **886**, Mars 2001.

[CC85] D. V. Chudnovsky and G. V. Chudnovsky. Applications of Padé approximations to the Grothendieck conjecture on linear equations. In *Lecture Notes in Mathematics, Number Theory*, vol. **1135** : 52-100, 1985.

[CK02] R. C. Churchill and J. J. Kovacic. Cyclic vectors. In *Differential Algebra and Related Topics, Proceedings of the International Workshop*, Rutgers University, Newark, November 2-3 2002.

[Cl03] T. Cluzeau. Factorization of differential systems in characteristic p. In *Proceedings of ISSAC'03*, ACM Press, 58-65, 2003.

[CH04] T. Cluzeau and M. van Hoeij. A modular algorithm for computing the exponential solutions of a linear differential operator. In *Journal of Symbolic Computation*, **38(3)** : 1043-1076, 2004.

[CE95] G. E. Collins and M. J. Encarnación. Efficient rational number reconstruction. In *Journal of Symbolic Computation*, **20(3)** : 287-297, 1995.

[CW04] É. Compoint and J.-A. Weil. Absolute reducibility of differential operators and Galois groups. In *Journal of Algebra*, **275** : 77-105, 2004.

[Co36] F. T. Cope. Formal solutions of irregular linear differential equations, Part II. In *Amer. J. Math.*, **58** : 130-140, 1936.

[Cor01] O. Cormier. On liouvillian solutions of linear differential equations of order 4 and 5. In *Proceedings of ISSAC'01*, ACM Press : 93-100, 2001.

[CSTU02] O. Cormier, M. F. Singer, B. T. Trager and F. Ulmer. Linear differential operators for polynomial equations. In *Journal of Symbolic Computation*, **34** : 355-398, 2002.

[DiV02] L. Di Vizio. Arithmetic theory of q-difference equations : the q-analogue of Grothendieck-Katz's conjecture on p-curvatures. In *Invent. Math.*, **150(3)** : 517-578, 2002.

[Du83] A. Duval. Lemmes de Hensel et factorisation formelle pour les opérateurs aux différences. In *Funkcial Ekvac.*, **26** : 349-368, 1983.

[En94] M. J. Encarnación. On a modular algorithm for computing gcds of polynomials over algebraic number fields. In *Proceedings of ISSAC'94*, ACM Press, 58-65, 1994.

[Ga66] F. R. Gantmacher. *Théorie des matrices, Tome 1.* Dunot (Paris), 1966.

[GG99] J. von zur Gathen and J. Gerhart. *Modern computer algebra.* Cambridge University Press, 1999.

[Gi98] M. Giesbrecht. Factoring in skew-polynomial rings over finite fields. In *Journal of Symbolic Computation*, **24(5)** : 463-486, 1998.

[GZ03] M. Giesbrecht and Y. Zhang. Factoring and decomposing Ore polynomials over $\mathbb{F}_p(t)$. In *Proceedings of ISSAC'03*, ACM Press, 127-134, 2003.

[GLR82] I. Gohberg, P. Lancaster and L. Rodman. Matrix polynomials. In Academic Press (New york, London, Paris), 1982.

[Gr90] D. Y. Grigoriev. Complexity of irreducibility testing for a system of linear ordinary differential equations. In *Proceedings of ISSAC'90*, ACM Press, 225-230, 1990.

[HS02] M. Hausdorf and M. Seiler. Involutives basis in MuPAD-Part I : involutive divisions. In *MathPad*, **11/1** : 51-56, 2002.

[Hoe98] S. Hoeppner. Differential equations with 2-term recursion. In *Manuscripta Math.*, **95(3)** : 349-362, 1998.

[Hon81] T. Honda. Algebraic differential equations. In *Symp. Math.*, **24** : 169-204, 1981.

[Ho96] M. van Hoeij. Rational solutions of the mixed differential equation and its application to factorization of differential operators. In *Proceedings of ISSAC'96*, ACM Press, 219-225, 1996.

[Ho97b] M. van Hoeij. Factorization of differential operators with rational functions coefficients. *Journal Of Symbolic Computation*, **24** : 537-561, 1997.

[Ho97a] M. van Hoeij. Formal solutions and factorization of differential operators with power series coefficients. *Journal Of Symbolic Computation*, **24** : 1-30, 1997.

[Ho98] M. van Hoeij. Rational solutions of linear difference equations. In *Proceedings of ISSAC'98* : 120-123, 1998.

[Ho99] M. van Hoeij. Finite singularities and hypergeometric solutions of linear recurrence equations. In *Journal of Pure and Applied Algebra*, **139** : 109-131, 1999.

[Ho02] M. van Hoeij. Factoring polynomials and the knapsack problem. *Journal of Number Theory*, **95** : 167-189, 2002.

[HM02] M. van Hoeij and M. Monagan. A modular gcd algorithm over number fields presented with multiple extensions. In *Proceedings of ISSAC'02*, ACM Press, 109-116, 2002.

[HRUW98] M. van Hoeij, J.-F. Ragot, F. Ulmer and J.-A. Weil. Liouvillian solutions of linear differential equations of order three and higher. In *Journal of Symbolic Computation*, **11** : 1-17, 1998.

[HW97] M. van Hoeij and J.-A. Weil. An algorithm for computing invariants of differential Galois groups. In *Journal of Pure and Applied Algebra*, **117-118** : 353-379, 1997.

[Ho81] T. Honda. Algebraic differential equations. In *Symp. Math.*, **24** : 169-204, 1981.

[In26] E. L. Ince. Ordinary differential equations. *Dover Publications*, INC. New York, 1926.

[IRS94] G. Ivanyos, L Rónyai, and A. Szántó. Decomposition of algebras over $\mathbb{F}_q(X_1, \ldots, X_m)$. In the journal *Appl. Alg. in Eng. Comm. and Comp.*, **5(2)** : 71-90, 1994.

[Ja37] N. Jacobson. Pseudo-linear transformations. In *Annals of Math.*, **38(2)** : 484-507, 1937.

[Ja85] N. Jacobson. *Basic algebra I, Second Edition.* W.H. Freeman and Compagny, New York, 1985.

[Ja80] N. Jacobson. *Basic algebra II.* W.H. Freeman and Compagny, San Francisco, 1980.

[Ja20] M. Janet. Sur les systèmes aux dérivées partielles. In *Journal de Math.*, 8-ème série, III, 65-151, 1924.

[Ja29] M. Janet. Leçons sur les systèmes d'équations aux dérivées partielles. In *Cahiers Scientifiques*, IV, Gauthiers-Villars, 1929.

[Ka70] N. Katz. Nilpotent connections and the monodromy theorem : applications of a result of Turritin. In *Publ. Math. I. H. E. S.*, **39** : 355-412, 1970.

[Ka82] N. Katz. A conjecture in the arithmetic theory of differential equations. *Bull. Soc. Math. France*, **110** : 203-239, 1982.

[LLL82] A. K. Lenstra, H. W. Lenstra Jr. and L. Lovász. Factoring polynomials with rational coefficients. In *Math. Ann.*, **261** : 515-534, 1982.

[Le94] A. H. M. Levelt. The semi-simple part of a matrix. In *Algorithmen in de Algebra*, 85-88, 1994.

[LST02] Z. Li, F. Schwarz and S. P. Tsarev. Factoring zero-dimensional ideals of linear partial differential operators. In *Proceedings of ISSAC'02*, ACM Press, 168-175, 2002.

[LST03] Z. Li, F. Schwarz and S. P. Tsarev. Factoring systems of linear PDEs with finite-dimensional solution spaces. In *Journal of Symbolic Computation*, **36(3-4)** : 443-471, 2003.

[Ma01] B. H. Matzat. Differential Galois theory in positive characteristic. Notes by J. Hartmann. In *IWR-Preprint*, 2001-35.

[MP02] B. H. Matzat and M. van der Put. Constructive differential Galois theory In *IWR-Preprint*, 2002-02.

[Mo04] M. Monagan. Maximal quotient rational reconstruction : an almost optimal algorithm for rational reconstruction. In *Proceedings of ISSAC'04*, ACM Press, 2004.

[MS98] T. Mulders and A. Storjohann. Fast algorithms for linear algebra modulo N. *Lecture notes in computer science*, **1461**, Algorithms-ESA'98 : 139-150, 1998.

[MS03] T. Mulders and A. Storjohann. On lattice reduction for polynomial matrices. *Journal of Symbolic Computation*, **35(4)** : 377-401, 2003.

[No29] N. E. Nörlund. Leçons sur les équations linéaires aux différences finies. In *Gauthiers Villard et Cie*, Paris, 1929.

[Or32] O. Ore. Formale theorie der linearen differentialgleichungen (Zweiter Teil), In *J. für d. Reine u. angew. Math.*, **168** : 233-252, 1932.

[Or33] O. Ore. Theory of non-commutative polynomials. In *Annals of Math.*, **34** : 480-508, 1933.

[Pe92] M. Petkovšek. Hypergeometric solutions of linear recurrences with polynomial coefficients. In *Journal of Symbolic Computation*, **14**(2-3) : 243-264, 1992.

[PWZ96] M. Petkovšek, H. S. Wilf and D. Zeilberger. $A = B$. With a foreword by Donald E. Knuth., 1996.

[Pf97] E. Pflügel. An algorithm for computing exponential solutions of first order linear differential systems. In *Proceedings of ISSAC'97*, ACM Press, 164-171, 1997.

[Pu95] M. van der Put. Differential equations in characteristic p. In *Comp. Math.*, **97** : 227-251, 1995.

[Pu96] M. van der Put. Reduction modulo p of differential equations. *Indag. Math.N.S.*, **7(3)** : 367-387, 1996.

[Pu97] M. van der Put. Modular methods for factoring differential operators. Unpublished manuscript (Preliminary Version), 1997.

[Pu01] M. van der Put. Grothendieck's conjecture for the rish equation $y' = ay + b$. *Indag. Math.*, **12(1)** : 113-124, 2001.

[PS97] M. van der Put and M. F. Singer. Galois theory of difference equations. In *Lectures Notes in Mathematics*, vol. **1666**, Springer (Berlin), 1997.

[PS03] M. van der Put and M. F. Singer. *Galois Theory of Linear Differential Equations*, vol. **328**. Grundlehren der mathematischen Wissenschaften, Springer, 2003.

[SSV04] D. Saunders, A. Storjohann and G. Villard. Matrix Rank Certification. In *Electronic Journal of Linear Algebra*, vol. **11** : 16-23, 2004.

[Sc89] F. Schwarz. A factorization algorithm for linear ordinary differential equations. In *Proceedings of ISSAC'89*, ACM Press, 17-25, 1989.

[Si91] M. F. Singer. Liouvillian solutions of linear differential equations with liouvillian coefficients. *Journal of Symbolic Computation*, **11** : 251-273, 1991.

[Si96] M. F. Singer. Testing reducibility of linear differential operators : a group theoretic perspective. In the journal *Appl. Alg. in Eng. Comm. and Comp.*, **7(2)** : 77-104, 1996.

[St03] A. Storjohann. High-order lifting and integrality certification. In *Journal of Symbolic Computation* **36(3-4)** : 613-648, 2003.

[To87] É. Tournier. Solutions formelles d'équations différentielles, Le logiciel de calcul formel : DESIR, Étude théorique et réalisation. Thèse de l'université de Grenoble, 1987.

[Ts94] S. P. Tsarëv. Some problems that arise in the factorization of linear ordinary differential operators. In *Programmirovanie*, **1** : 45-48, 1994. Translation in *Programming and Comput. Software*, **20(1)** : 27-29, 1994.

[Ts98] S. P. Tsarëv. Factorization of linear partial differential operators and the Darboux integrability of nonlinear PDEs. In *SIGSAM Bulletin*, **32(4)** : 21-28, 1998.

[Ts01] S. P. Tsarëv. Factorization of overdetermined systems of linear partial differential equations with finite-dimensional solution space. In *Computer algebra in scientific computing* (Konstanz) : 529-539, 2001.

[Ul03] F. Ulmer. Liouvillian solutions of third order differential equations. In *Journal of Symbolic Computation*, **36** : 855-889, 2003.

[UW96] F. Ulmer and J.-A. Weil. Note on Kovacic's algorithm. In *Journal Of Symbolic Computation*, **22** : 179-200, 1996.

[We01] C. Weixlbaumer. Solutions of difference equations with polynomial coefficients. Diplomarbeit, RICS Linz, Johannes Kepler Universität, 2001.

[Wu04] M. Wu. Factorization and decomposition of linear functional systems. PhD thesis, in preparation, 2004.

Abstract : Modular methods lead to very efficient algorithms in many areas of computer algebra and particularly for the study of algebraic equations. The goal of this thesis is to show how these modular techniques can be adapted to the differential case and allow to create new algorithms (or to improve existing algorithms) for linear differential equations. The first part deals with the factorisation of differential operators in positive characteristic. The "miracle" in characteristic p is that the problem reduces to linear algebra. Using this fact, we develop algorithm for factoring differential systems. We give the complexity of the distinct steps of this algorithm. Finally, we generalize it to the setting of partial differential systems. The topic of the second part is making Beke's algorithm to compute the exponential solutions of linear differential equations more efficient. This algorithms suffers from two drawbacks : a combinatorial problem and a field problem. We show that combining local "geometric" data (the generalized exponents) and modular "arithmetic" data (the eigenvalues of the p-curvature) allows to decrease the number of combinations usually considered by the algorithm and to reduce the degree of the algebraic extensions of the base field needed to compute the exponential solutions. In the third part, we prove that a similar approach applies to the same problem for difference equations. In the last section, we develop a fully modular algorithm for computing the polynomial solutions of linear differential equations in characteristic zero. We evaluate the relevance of the modular informations that can be obtained for this problem. We give and compare the complexity of our algorithm and that of the existing ones. Finally, thanks to experimental comparisons, we exhibit a class of differential equations for which our modular approach is more relevant than existing algorithms. Most of our algorithms have been implemented in the computer algebra system Maple.

Résumé : Les méthodes modulaires conduisent à des algorithmes très efficaces dans de nombreux domaines en calcul formel et notamment dans celui des équations algébriques. Le but de cette thèse est de montrer comment ces techniques modulaires s'adaptent au cas différentiel et permettent de développer de nouveaux algorithmes (ou d'améliorer des algorithmes existants) pour l'étude d'équations différentielles linéaires. La première partie traite du problème de la factorisation d'opérateurs différentiels en caractéristique positive. Le « miracle » de la caractéristique p est que le problème peut se réduire à de l'algèbre linéaire. En exploitant ce fait, nous développons un algorithme de factorisation de systèmes différentiels. Nous donnons la complexité des différentes étapes de cet algorithme. Enfin, nous le généralisons au cadre de systèmes d'équations aux dérivées partielles. L'objet de la deuxième partie est de rendre plus efficace l'algorithme de Beke pour le calcul des solutions exponentielles d'équations différentielles linéaires. Cet algorithme possède deux inconvéniants majeurs qui le rendent peu efficace : un problème combinatoire et un problème de corps. Nous montrons qu'en combinant des informations « géométriques » locales (les exposants généralisés) et des informations « arithmétiques » modulaires (les valeurs propres de la p-courbure), nous pouvons diminuer le nombre de combinaisons considérées habituellement par l'algorithme et réduire le degré des extensions algébriques du corps de base nécessaires au calcul des solutions exponentielles. Dans la troisième partie, nous démontrons qu'une démarche similaire s'applique pour le problème analogue dans le cas des équations aux différences. Finalement, dans la dernière partie, nous développons un algorithme entièrement modulaire calculant les solutions polynomiales d'équations différentielles linéaires en caractéristique zéro. Nous évaluons la pertinence des informations modulaires que l'on peut obtenir pour ce problème. Nous donnons et comparons les complexités de notre algorithme et des algorithmes existants. Puis, grâce à des comparaisons expérimentales, nous exhibons des classes d'équations pour lesquelles notre approche modulaire est mieux adaptée que les algorithmes existants. La plupart de nos algorithmes ont été implantés dans le logiciel de calcul formel Maple.